A publication of ISCA*:
International Society for Computers
and Their Applications

INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS

TABLE OF CONTENTS

***"International Journal of Computers and Their Applications is Peer Reviewed".**

Volume 29, No. 2, June 2022 **ISSN 1076-5204**

International Journal of Computers and Their Applications

A publication of the International Society for Computers and Their Applications

ISCA Headquarters···•···278 Mankato Ave, #220, Winona, MN 55987···•···Phone: (507) 458-4517
E-mail: isca@ipass.net • URL: http://www.isca-hq.org

Guest Editorial

This issue of IJCA is a collection of six refereed papers, four of which are selected from the 34th International Conference on Computer Applications in Industry and Engineering (CAINE 2021). The other two papers in this issue are regular submissions to IJCA.

Each paper submitted to this issue was reviewed judging the originality, technical contribution, significance and quality of presentation.

The papers in this issue cover a wide range of research interests in the community of computers and applications. The topics and main contributions of the papers are briefly summarized below.

Antoine Bossard of Kanagawa University, Japan, proposed in their paper "*Proposal and Evaluation of a Chinese Character Hash Function Based on Strokes for Fingerprinting*" practical approaches a novel hash function for Chinese character encoding. The proposed hash function is solely based on the strokes of the character to make it non-ambiguous. The function with low collision rate was evaluated both theoretically and in practice to show its validity and applicability.

Robert N. K. Loh and K. C. Cheok of Oakland University, USA, in the paper "*Analysis and Control of Linear Time-Varying (LTV) Systems*" investigated the problem of determining analytical solutions for the fundamental and state transition matrices associated with linear time-varying (LTV) systems. It provided a thorough theoretical analysis as well as examples to demonstrate the analysis.

Lee A. Belfore of Old Dominion University, USA, proposed a logic level modeling framework for adiabatic circuit operation using the Very High Speed Integrated Circuit Hardware Description Language (VHDL), in the paper "*Logical Modeling of Adiabatic Logic Circuits using VHDL with Examples*." Three modeling examples with simulation results were given to demonstrate the use of the framework.

Thomas Bidinger, Hannah Buzard, James Hearne, Amber Meinke and Steven Tanner of Western Washington University, USA, proposed an algorithm by exploring the theory of causal regularity and applied it to a number of real-world data sets to identify causal relationships in them, in the paper "*Mining for Causal Regularities*." Experiments on several data sets showed the proposed algorithm could successfully identify singular and conjunctive conditions that serve as possible causes for a chosen event.

Shuvo Kumar Paul, Arjun Vettath Gopinath, Mircea Nicolescu, and Monica Nicolescu of University of Nevada, Reno, USA, and Pourya Hoseini of University of California, San Diego, USA, in the paper "*Integration of Multimodal Inputs and Interaction Interfaces for Generating Reliable Human-Robot Collaborative Task Configurations*" addressed reliability and trust problems in human-robot interaction with a proposed framework that integrates pointing gesture and speech with sensor input to generate reliable task configurations for human-robot collaborative environment.

Narayan C. Debnath, Archana Patel, Debarshi Mazumder, Phuc Nguyen Manh, and Ngoc Ha Minh of Eastern International University, Vietnam, presented an ontology evaluation framework that is an integration of four known approaches to deal with different criteria of evaluation in their paper "*InFra_OE: An Integrated Framework for Ontology Evaluation*." The framework may overcome some shortcomings of these individual methods.

As guest editors we would like to express our appreciation of the contributions of the authors, as well as the experts who reviewed all the papers submitted to this issue. We also thank the help and support from Dr. Ajay Bandi, the editor-in-chief of IJCA.

We hope you enjoy this issue of IJCA. More information about the ISCA society can be found at http://www.isca-hq.org.

Guest Editors

Gongzhu Hu, Central Michigan University, USA
Yan Shi, University of Wisconsin Platteville, USA
Takaaki Goto, Toyo University, Japan

June 2022

Proposal and Evaluation of a Chinese Character Hash Function Based on Strokes for Fingerprinting*

Antoine Bossard†
Kanagawa University
Tsuchiya 2946, Hiratsuka, Kanagawa 259-1293, JAPAN

Abstract

Chinese character representation in computer systems has been a long-standing issue, which is directly related to the information representation and character encoding fields of computer science. For example, as of today some Chinese characters still cannot be easily input in a computer, let alone be universally represented (identified). In this research, we have been especially focusing on such Chinese characters that are not covered by the conventional character encodings, and in this paper, after having previously introduced a universal character encoding for Japanese, we propose a non-ambiguous hash function applicable to any Chinese character. Unlike previous and related works, the proposed function is solely based on the strokes of the character, thus leaving no room for ambiguity. Considering the sparsity and the low collision rate of the described hash function, fingerprinting is a meaningful application, which can then be used for information retrieval purposes, among others. Let us emphasize that simplicity and unambiguity are the two keys of this proposal. The described character hashing method is then evaluated both theoretically and in practice in order to quantitatively show its validity, applicability and contribution.

Key Words: Japanese; Chinese; *kanji*; character; symbol.

1 Introduction

Because Chinese characters are tens of thousands, their representation and processing in general has always been a difficult issue for computer systems. Several approaches were considered as hardware and software evolved over the years. In the early days of computing, the characters were hard-coded into read-only memory (ROM), a solution that is interestingly still in use today for example with some LCD panels [8]. Due to the lack of flexibility of this ROM approach, logical encodings such as those defined by the Japanese Industrial Standards Committee (JISC) rapidly replaced it.

There are two main approaches for such logical character encodings: the unifying approach which considers all the writing systems at once, followed for instance by Unicode [16], and the non-unifying approach which includes the encodings that are local to one writing system, such as Shift-JIS for Japanese. Because both of these two approaches fail at addressing the representation of *any* Chinese character, we recently proposed a universal character encoding for Japanese (UCEJ) [5], which is based on a three dimensional space used to assign distinct coordinates to characters. As per the definition of UCEJ, the first coordinate identifies the character radical, the second coordinate holds the number of strokes of the character and the third one distinguishes variants (forms) of a same character.

It is well known that some Chinese characters are morphologically structured according to composition patterns, such as vertical and horizontal combinations [2, 3, 7, 15]. Such (de)composition patterns can sometimes be ambiguous (e.g. the definition of the support set $\tilde{R}$ of [2]), and moreover they cannot be applied directly to character strokes since strokes, unlike characters, are not structured according to easily identifiable patterns. Hence, this decomposition approach is not a solution to unambiguous character representation.

Besides, a conventional character encoding like Unicode cannot be used as a hashing function given that only a limited number of characters are covered: many Chinese characters are left unsupported, that is unassigned to a code point, and this is typically the case of Chinese characters that are local to one culture, such as the *kokuji* characters of Japanese [2]. Because aimed at supporting any character, the conventional encoding UCEJ could be considered for hashing and fingerprinting, but since the hash value (coordinate) that corresponds to a character cannot be fully deterministically calculated, this is not a solution either.

In continuation of UCEJ, the objective of this research is the proposal of a non-ambiguous hash function that can be applied to *any* Chinese character. Identification of any Chinese character in a unique and unambiguous manner is a first application. Given in some cases the existence of numerous variants for a same character – regularly excluded from conventional encodings – this is a far from trivial issue. Hence, calculating a Chinese character fingerprint is meaningful to refer to a character that is absent from conventional encodings.

The rest of this paper is organized as follows: hashing,

*This paper is an extended version of [1].

†Graduate School of Science. Email: abossard@kanagawa-u.ac.jp.

fingerprinting and character properties are briefly recalled in Section 2. The proposed hash function is then presented in Section 3. It is next theoretically and practically evaluated in Sections 4 and 5, respectively. Section 6 concludes this paper.

2 Preliminaries

We first make a brief recall regarding hashing. Hash functions are frequently encountered in computer science: they are used to calculate an index from a datum so that this datum can be used to directly refer to, for example, the corresponding entry in a table in memory. In the case of table indices, the calculated values are expected to fall within a range so that the table entries tend to be consecutive in memory, and this without any assumption on the sizes of the original data. Besides, the indices calculated for distinct data are expected to be distinct too. If they are not, we say that collisions occur [12].

Such a function which realizes a mapping between data and identifiers, like indices, has other applications, for instance fingerprinting: rather than calculating consecutive or near consecutive table indices, a fingerprint is typically used to identify a datum of arbitrary size, and this with a more or less short value. This is comparable to the scientific applications of human fingerprints. The algorithm described by Rabin is a classic fingerprinting example [6].

Figure 1: *taito*

Next, we recall essential properties of the Chinese characters. Each Chinese character has one radical, although there exist some characters for which the radical is not clearly identified, or at least is still debated (this is especially the case for characters that have undergone simplifications [2]). Each character has at least one reading, although there are usually several, especially when various languages whose writing system involves Chinese characters are considered.

A character is made of strokes (calligraphic brush strokes), and there is a consensus that the highest number of strokes in a Chinese character, at least in Japanese, is 84. This character, illustrated in Figure 1, is the *taito* character (a.k.a. *daito*, *otodo*) [9]. Furthermore, the strokes of a character are drawn in a precise order, although this order may depend on the writing system considered [2]. Besides, it should be noted that a character can have variants, which are in some cases numerous [13]. Additional details can be found for example in [14].

3 Methodology

We describe in this section the proposed hash function. This function is solely based on character strokes: it relies on the stroke number, the stroke types and the stroke writing order. Because it is essential, we emphasize here that this approach to the function definition induces no ambiguity at all. For comparison, we relied in previous researches for character processing on character radicals and character decomposition operations, two properties which are more (the latter) or less (the former) ambiguous. As recalled in Section 2, the number of strokes, the types of the strokes and the writing order of the strokes for a Chinese character is indeed unambiguously defined. Even if the writing order of the character strokes may differ for a few characters from one writing system to another, such as between Japanese and Chinese, it is clearly defined when considering one writing system. For example, the stroke order of the Chinese characters used in Japanese has been formally established by the Japanese government [10].

So as to lower the collision probability, the proposed function involves all the three aforementioned stroke properties: stroke number, stroke types and stroke order. Regarding stroke types, 36 strokes have been defined by the Unicode consortium for Chinese characters: this is the 31C0–31EF code block [16]. These 36 strokes are shown in Table 1; we have assigned to each of them (columns labeled "Str.") a unique identifier (columns labeled "Id.").

Table 1: The 36 strokes for Chinese characters (Unicode block 31C0–31EF). They are each assigned a unique identifier

Id.	Str.	Id.	Str.	Id.	Str.	Id.	Str.	Id.	Str.	Id.	Str.
0	㇀	6	㇆	12	㇋	18	㇒	24	㇟	30	㇡
1	㇁	7	㇇	13	㇌	19	㇓	25	㇠	31	㇢
2	㇂	8	㇈	14	㇡	20	㇔	26	㇣	32	㇣
3	㇃	9	㇉	15	㇍	21	㇕	27	㇤	33	㇥
4	㇄	10	㇊	16	㇐	22	㇖	28	㇦	34	㇧
5	㇅	11	㇎	17	㇑	23	㇗	29	㇩	35	㇪

Define S the set of these 36 character strokes, and $k : S \to \{0,1,\ldots,35\}$ the bijection between a stroke and its identifier. Let C be the set of Chinese characters; it is recalled that its cardinality is unknown. For a character $c \in C$ of $n \in \mathbb{N}^*$ strokes $s_i \in S$ $(0 \leq i \leq n-1)$ and of stroke order that induced by the relation $i < j \Rightarrow s_i$ written before s_j $(0 \leq i, j \leq n-1)$, we define the hash function h as follows:

$$\begin{array}{rcccl} h & : & C & \to & \mathbb{N} \\ & & c & \mapsto & \displaystyle\sum_{i=0}^{n-1} 2^{6i}\, k(s_i) \end{array}$$

In other words, stroke identifiers are each represented with six bits, and the stroke number as well as the stroke order are directly induced by the concatenation of 6-bit sequences. The fingerprint can thus be conveniently represented with the octal notation: each stroke corresponds to two octal digits. Examples of fingerprint calculations are given in Table 2; in this table, the stroke order is indicated from left to right and fingerprints are given in the octal notation, with the most significant digit on the left.

Once a fingerprint has been obtained, it can then be adjusted for hashing purposes (e.g. hash table), that is, to reduce the

Table 2: Fingerprint calculation for sample Chinese characters

Character	Stroke number	Stroke types and stroke order	Fingerprint (octal notation)
大 "large"	3	一, 丿, 乀	17 22 20
水 "water"	4	亅, 𠂆, 丿, 乀	17 22 07 32
凧 "kite"	5	丿, 乙, 丨, 冂, 丨	21 06 21 10 22
迄 "until"	7	丿, 一, 乙, 丶, 丶, 𠂇, 乀	17 13 24 24 40 20 22

sparsity of the obtained fingerprints. This would be at the cost of an increased collision rate though. For example, hashing with folding by summing each stroke value, or division hashing by applying a modulo function to the obtained fingerprints.

4 Theoretical Evaluation: Size and Sparsity

4.1 Memory Size Requirements

First, let us compare the size of fingerprints versus the size of a character coordinate in UCEJ. To this end, we first recall that each character stroke is represented on 6 bits, and that the highest number of strokes in a Chinese character, at least in Japanese, is 84 is a consensus. So, a character of n strokes requires at most $6n$ bits ("at most" because the last stroke may not require all the six bits, thus resulting in a few zeros at the MSB, in other words digits that can be discarded). So, an n-stroke character is expressed on at most $6n/8 = 0.75n$ bytes. On the other hand, the coordinate of any character in UCEJ takes 10 bytes [5]: the required memory size does not depend on the character. And in the case of the refinement of UCEJ which takes into account stroke types and the stroke order, each character coordinate takes 38 bytes, again no matter the character [4]. This memory size requirement comparison is illustrated in Figure 2; because a conventional encoding such as Shift-JIS or Unicode only supports a fraction of the Chinese characters, it is not included in this comparison as it would be obviously largely unfair. Given that the vast majority of Chinese characters have at most 30 strokes (this is further detailed in Section 4.2 below), the memory size taken by a fingerprint remains reasonable compared to a UCEJ coordinate.

It is however critical to note that a UCEJ coordinate cannot be completely calculated from a character: as recalled in the introduction, the UCEJ lookup function calculates from a character its X and Y coordinates only, thus not involving Z. This is a major drawback compared to the fingerprint calculation method proposed herein, and one reason for that lookup function not being a suitable hashing function.

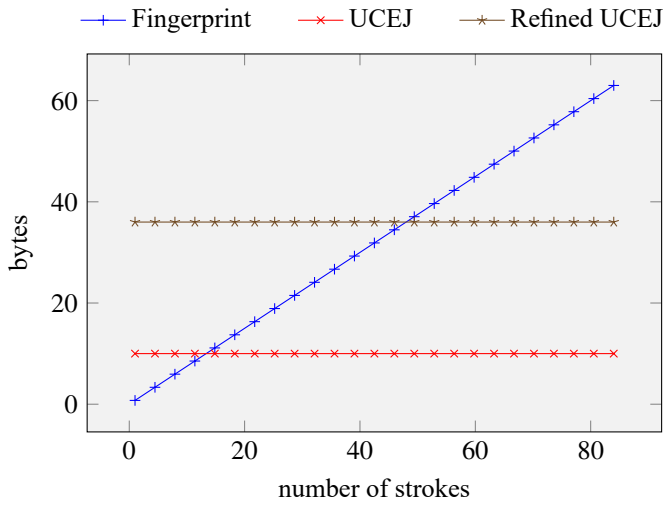


Figure 2: Memory size requirement of a fingerprint versus a UCEJ coordinate

4.2 Hash Function Sparsity

Next, we analyze the projected sparsity of the calculated fingerprints. Directly from above, we have that the fingerprint of a 1-stroke character is in the interval $[0, 2^6 - 1]$ (since six bits per stroke), that of a 2-stroke character in the interval $[2^6, 2^{12} - 1]$ (since twelve bits for the two strokes) and so on. Because a character includes at most 84 strokes as recalled, a fingerprint consists in at most $84 \times 6 = 504$ bits. Therefore, there are a total of 2^{504} distinct fingerprints, which is of course significantly larger than the number of Chinese characters (even if only an estimation, several tens of thousands, of this character grand total is known). So, the character density in the range of the possible fingerprint values is globally low.

The distribution of the stroke number of the Chinese characters used in Japanese is illustrated in Figure 3. For reference, we have represented in the same plot the maximum number of bits required to represent the fingerprint of a character depending on the stroke number. These data have been extracted from the List of MJ Characters provided by the Japanese Character Information Technology Promotion Council [11]. This database includes in total 58 862 characters. Note that 84 has been considered as the highest stroke number as explained, but since the *otodo* character does not appear in the database, the number of occurrences therein is 0. Hence, although this database is rather exhaustive, the zero number of occurrences as soon as stroke number 65 is yet another indicator of the lacking support of the Chinese characters by computer systems.

It should be noted that the proposed fingerprinting algorithm is not perfect in the sense that it is possible – although rather rare – to find two distinct characters that induce the same fingerprint, for example 引 *hiku* and 弔 *tomurau*, both of fingerprint 21

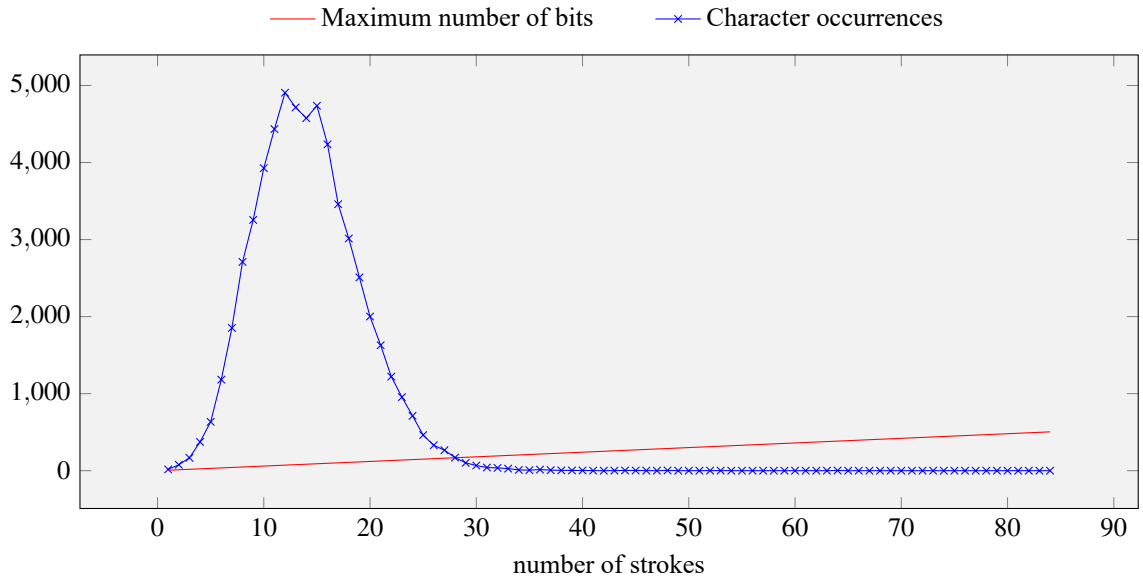


Figure 3: Distribution of the stroke number of the Chinese characters used in Japanese

`11 20 25` (octal notation). In other words, the described hashing function is not injective. In an attempt to further reduce the collision rate, additional character properties could be considered. Nonetheless, this would be at the cost of increased ambiguity in the function definition. It is recalled that we have completely eliminated such ambiguity with the approach proposed in this paper. Besides, in this search for a perfect hashing function, it will become clear that the successively established functions, defined at the beginning in a discrete manner, will inevitably evolve towards a continuous (i.e. non-discrete) function, which is problematic considering the hashing applications.

Finally, it is interesting to remark the following paradox regarding character density: the characters that have the greatest stroke number are those whose fingerprint occupies the greatest number of bits but which are the least "dense" characters. That is, when considering characters of at most n strokes, the number of representable such characters is 2^{6n}, but at the same time as n increases, the number of n-stroke characters (i.e. character occurrences) decreases. This is clearly visible in Figure 3.

5 Practical Evaluation: Collision Analysis

5.1 Methodology

In this section, we conduct another quantitative analysis by measuring in practice the collision rate of the proposed hash function. To this end, we have generated a database that associates to a Chinese character the ordered sequence of its strokes, such including the stroke types and stroke order information.

This database is essential to this work; it has been created with the following recursive algorithm.

Step 1. We have manually defined character (topmost) decompositions: only the vertical and horizontal composition operations have been considered, which is not an issue since these two composition operations cover the vast majority of Chinese characters (more than 80% for a representative character subset [2]).

For instance, the topmost decomposition operation of the character 加 is defined as the algebraic expression 力 + 口, with "+" the horizontal composition operation.

Step 2. We have manually defined the ordered stroke sequence for a few characters that are "prime", that is which cannot be further decomposed [2]. This is typically the case for character radicals. This step induces the base case of the recursion.

For instance, the ordered stroke sequence for the character radical 禾 is manually defined as: ノ, 一, 丨, ノ, 乀.

Step 3. For each character c of the database obtained at Step 1, we recursively calculate its ordered stroke sequence as follows: if a stroke sequence for c has been already defined (i.e. during Step 2 or Step 3), it is returned. Otherwise, let $c_1 \bullet c_2$ be the decomposition of c obtained from the database of Step 1, with "$\bullet$" a composition operation such as "+". We apply this process recursively on c_1 and c_2 to obtain the ordered stroke sequences $\bar{c}_1$ and $\bar{c}_2$, respectively. Let $\bar{c}$ be the concatenation of the two ordered stroke sequences $\bar{c}_1$ and $\bar{c}_2$. We record and return this newly obtained ordered stroke sequence $\bar{c}$ for the character c.

This stroke sequence calculation method is exemplified below. For instance, consider $c =$ 量. No stroke sequence exists for this character. Its decomposition 旦 × 里 is obtained from the database of Step 1, with "×" the vertical composition operation.

Next, we have $c =$ 旦. No stroke sequence exists for this

character. Its decomposition 日 × 一 is obtained from the database of Step 1.

So, we have $c =$ 日. This character is prime, and thus its ordered stroke sequence has already been calculated (Step 2); it is returned: 丨, 冂, 一, 一. Then, we have $c =$ 一. This character is also prime, and thus its ordered stroke sequence has already been calculated (Step 2); it is returned: 一.

Hence, the ordered stroke sequence for $c =$ 旦 is obtained by concatenation: 丨, 冂, 一, 一, 一. Next, we have $c =$ 里. This character is prime, and thus its ordered stroke sequence has already been calculated (Step 2); it is returned: 丨, 冂, 一, 一, 丨, 一, 一. Therefore, the ordered stroke sequence for $c =$ 量 is obtained by concatenation: 丨, 冂, 一, 一, 一, 丨, 冂, 一, 一, 丨, 一, 一.

It is important to note that this newly created stroke sequence database is not perfect: for example, we assume that the stroke order for a character decomposed as $c_1 \bullet c_2$ consists first of the strokes of c_1 and then of those of c_2. This is true in most cases, but there can be exceptions, albeit rare. In addition, some stroke sequences are assumed, like 丶, 丶, 𠂉 (i.e. 氵) for 水. That is, when the character 水 is encountered, its stroke sequence is assumed to be that of its variant 氵; in this research, this variant 氵 largely supersedes 水 so this assumption is safe.

Also, all the decomposition operations identified for Chinese characters (refer to [2]) are not present in the decomposition database (i.e. the first step of the algorithm), so the calculated fingerprints are for a part of the characters. But as recalled previously, the horizontal and vertical decomposition operations cover the vast majority (80%) of characters. Hence, this suffices to obtain representative fingerprints.

5.2 Results

We give in this section raw results of the collision analysis experiment. These results are discussed in the next section. First, the distribution of the calculated ordered stroke sequences based on the stroke number is shown in Figure 4.

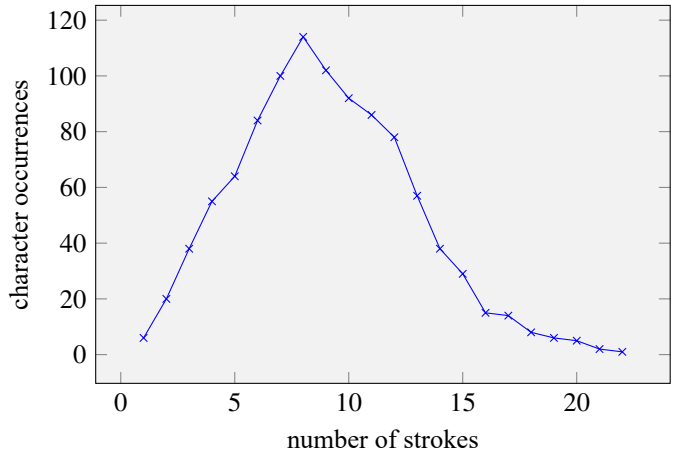


Figure 4: Distribution of the calculated ordered stroke sequences based on the stroke number

Next, we quantitatively measure collisions: 46 characters have not a unique ordered stroke sequence, and have thus not a unique fingerprint. These characters for which hash collisions would occur are summarized in Table 3. In this table, characters are grouped so that the character of a same group have the same ordered stroke sequence. The stroke number is also given for reference. The groups marked with an asterisk (*) need additional explanations: they are given in the next section.

Table 3: A summary of the detected collisions, that is, characters which have a non-unique ordered stroke sequence. Characters are grouped when they have the same ordered stroke sequences

Character group	Strokes	Character group	Strokes
人, 八	2	吉, 吉	6
力, 刀	2	伝, 会 *	6
土, 工, 士	3	隶, 彔 *	6
彳, 心 *	3	貝, 昊	7
日, 曰	4	呈, 里	7
太, 犬	4	昌, 胆	8
六, 文	4	径, 怪	8
公, 仏 *	4	治, 冶 *	8
肉, 月 *	4	査, 相	9
召, 加	5	唄, 員	10
旦, 目, 且	5	准, 淮 *	11

Finally, we have extracted from the realized character stroke database the occurrence frequency of each of all the stroke types (refer to Table 1). The details of this analysis are given in Table 4. In this table, the stroke identifier (“Id.”), visual rendering (“Str.”), number of occurrences (“Occur.”) and percentage are given for each detected stroke.

Table 4: Frequency of the stroke types as calculated from the realized database

Id.	Str.	Occur.	%	Id.	Str.	Occur.	%
16	一	2 769	30.28	31	乚	100	1.09
17	丨	1 565	17.12	22	乛	95	1.04
18	丿	1 245	13.62	25	亅	58	0.63
20	丶	1 004	10.98	2	乀	33	0.36
21	冂	546	5.97	8	乙	29	0.32
15	乀	518	5.66	4	𠃌	26	0.28
6	冖	216	2.36	23	𠃍	21	0.23
26	亅	205	2.24	12	𠃋	16	0.17
19	丿	182	1.99	1	𠃊	12	0.13
28	𠂉	180	1.97	27	𠂆	12	0.13
0	𠂇	176	1.92	9	𠂊	5	0.05
7	𠂇	127	1.39	30	𠂊	4	0.04

5.3 Discussion

Ordered stroke sequences for a total of 1 014 characters were successfully calculated. The distribution of the calculated ordered stroke sequences according to the stroke number shown in Figure 4 shows that the characters for which stroke sequences were calculated have a distribution that is a on a par with Chinese characters in general (see Figure 3) and thus the validity (generality) of the results obtained in this experiment.

The collision analysis shows that only 46 characters that induce collisions were found, and this in 22 character groups. In other words, this is a collision rate of approximately 4.5%.

However, it should be noted that 14 out of these 46 characters are false positives (they are marked with an asterisk in Table 3): for example, the collisions induced by the two character pairs (准, 淮) and (治, 冶) are false positives: they are detected as collisions because of database assumptions (precisely, that 水 is assumed to be of the form 氵), and will thus not induce collisions in practice. This is also the case for the character pairs (肉, 月), (彳, 心), (公, 仏) and (伝, 会): they are found to induce collisions as well since we assumed 肉 to be of the form 月, 心 to be of the form 忄 and 亻 to be of the form 人 for the same reason we assumed 水 to be of the form 氵. Finally, this is also the case for the pair 隶, 彔: the character element 彐 was assumed as a simplification of character strokes. Hence, the collision rate in this experiment actually amounts only to 3.2% (i.e. 32 characters).

Regarding the stroke type analysis, in total 9 144 strokes were enumerated, and several stroke types were absent from the database: 24 stroke types were detected out of the total 36 (see Table 1). The frequency of the horizontal stroke 一 (more than 30%) is significantly higher than that of the rest. It is not a surprising result since the fact that this stroke is also both a radical and a character (一) shows the omnipresence of this brush drawing.

6 Conclusions

The processing, let alone representation, of Chinese characters in computer systems has been a long-standing issue. It is, for example, still not possible to input some characters into systems, albeit infrequently used ones. To tackle this problem, we have recently introduced a universal character encoding for Japanese (UCEJ). However, UCEJ still lacks a fully deterministic way of calculating the code point of a character. Directly related to this issue, in this paper we have described a non-ambiguous hashing function for fingerprinting Chinese characters. One objective of this work is to facilitate the identification and processing in general of Chinese characters by computer systems. The proposed hashing method has been both theoretically and practically evaluated so as to quantitatively show its validity, applicability and contribution.

As for future works, refining the function definition in an attempt to further reduce the collision probability of the computed fingerprints is a meaningful objective. This however involves several issues, like the sparsity of the hash values, the collision rate, and the simplicity and discreteness of the established function, and because they are interdependent there is no other way but to consider them simultaneously.

References

[1] Antoine Bossard. “A Chinese Character Hash Function Based on Strokes for Fingerprinting.” *Proceedings of the 34th International Conference on Computer Applications in Industry and Engineering (CAINE; 11–13 October, online)*, *EPiC Series in Computing*, 79:64–70, 2021.

[2] Antoine Bossard. *Chinese Characters, Deciphered*. Kanagawa University Press, Yokohama, Japan, March 2018.

[3] Antoine Bossard and Keiichi Kaneko. “Chinese Characters Ontology and Induced Distance Metrics.” *International Journal of Computers and Their Applications*, 23(4):223–231, 2016.

[4] Antoine Bossard and Keiichi Kaneko. “Refining the Unrestricted Character Encoding for Japanese.” *Proceedings of 34th International Conference on Computers and Their Applications (CATA; 18–20 March, Honolulu, HI, USA)*, *EPiC Series in Computing*, 58:292–300, 2019.

[5] Antoine Bossard and Keiichi Kaneko. “Unrestricted Character Encoding for Japanese.” *Databases and Information Systems X*, *Frontiers in Artificial Intelligence and Applications*, 315:161–175, January 2019.

[6] Andrei Z. Broder. “Some Applications of Rabin’s Fingerprinting Method.” *Sequences II*, pp. 143–152, 1993.

[7] Osamu Fujimura and Ryohei Kagaya. “Structural Patterns of Chinese Characters.” *Proceedings of the Conference on Computational Linguistics (1–4 September, Sånga-Säby, Sweden)*, pp. 1–17, 1969.

[8] Hitachi, Tokyo, Japan. *HD44780U (LCD-II) (Dot Matrix Liquid Crystal Display Controller/Driver)*, ADE-207-272(Z), ’99.9, Rev. 0.0, 1998.

[9] Takehiro Ito. “辞書になかった最多画数の漢字「幽霊文字」の怪…「タイト」さんをご存じないですか?” *The Yomiuri Shimbun (online)*, November 2020. `https://www.yomiuri.co.jp/life/20201030-OYT8T50053/` In Japanese. Last accessed June 2022.

[10] Japanese Ministry of Education, Science, Sports and Culture (文部省). 筆順指導の手びき. First Edition. In Japanese, March 1958.

[11] Japanese Character Information Technology Promotion Council (一般社団法人 文字情報技術促進協議会). List of MJ characters (MJ文字情報一覧表). `https://moji.or.jp/mojikiban/mjlist/` Version 006.01. In Japanese, May 2019. Last accessed June 2022.

[12] Donald E. Knuth. *The Art of Computer Science – Volume 3*, Second Edition. Addison-Wesley, Boston, MA, USA, 1998.

[13] Kyoo-Kap Lee. "Causes of Variant Forms as a Result of Structural Changes to Character Components." *Journal of Chinese Writing Systems*, 1(1):29–35, 2017.

[14] Ken Lunde. *CJKV Information Processing*, Second Edition. O'Reilly Media, Sebastopol, CA, USA, 2009.

[15] Richard Sproat. *A Computational Theory of Writing Systems*. Cambridge University Press, Cambridge, England, 2000.

[16] The Unicode Consortium. *The Unicode Standard 5.0*. Addison-Wesley, Boston, MA, USA, 2007. More recent versions accessible online at `http://www.unicode.org/versions/latest/` Last accessed June 2022.

Antoine Bossard received the B.S. and M.S. degrees from Université de Caen Basse-Normandie, France in 2005 and 2007, respectively, and the Ph.D. degree from Tokyo University of Agriculture and Technology, Japan in 2011.

He is an Associate Professor of the Graduate School of Science, Kanagawa University, Japan. His research is focused on graph theory, interconnection networks, and dependable systems. For several years, he has also been conducting research regarding Chinese characters and their processing by computer systems. He is a member of ACM, ACIS, ISCA, and TUG.

Analysis and Control of Linear Time-Varying (LTV) Systems

Robert N. K. Loh* and K. C. Cheok*
Oakland University, Rochester, MI, 48309-4401, USA

Abstract

Consider a linear time-varying (LTV) system described by the state-space equation $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u}(t)$. The main objectives of this paper include (**i**) determination of the analytical or closed-form solutions for the fundamental matrix $\mathbf{X}(t)$ and the state transition matrix $\mathbf{\Phi}(t, t_o)$ of the LTV system; (**ii**) design of feedback control, such that the closed-loop system matrix $\mathbf{A}_{cl}(t) = \mathbf{A}(t) - \mathbf{B}(t)\mathbf{K}(t)$, where $\mathbf{K}(t)$ is a gain matrix, has desirable properties, in particular, $\mathbf{A}_{cl}(t)$ being commutative and triangular; and (**iii**) design of observers such that the observer matrix $\mathbf{A}_o(t)=\mathbf{A}(t)-\mathbf{H}(t)\mathbf{C}(t)$, where $\mathbf{H}(t)$ is the observer gain matrix, has desirable properties as in (**ii**), namely, $\mathbf{A}_o(t)$ being commutative and triangular. The commutativity and triangularization of $\mathbf{A}_{cl}(t)$ and $\mathbf{A}_o(t)$ facilitate the analytical solutions for their fundamental and state transition matrices. Examples and simulations demonstrate the design objectives.

Key Words: Linear time-varying (LTV), feedback, controller, observer, commutativity, triangular, separation principle, triangulatiaon, commutativity.

1 Linear-Time-Varying (LTV) Systems

Consider an nth-order linear time-varying (LTV) system described by the state-space equation:

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u}(t), \quad \mathbf{x}(t_o) = \mathbf{x}_o, \tag{1a}$$

$$\mathbf{y} = \mathbf{C}(t)\mathbf{x}, \tag{1b}$$

where $\mathbf{x}(t)$ is an $n\times 1$ state vector, $\boldsymbol{u}(t)$ an $\ell\times 1$ control vector, and $\mathbf{y}(t)$ an $m\times 1$ output vector; $\mathbf{A}(t)$, $\mathbf{B}(t)$ and $\mathbf{C}(t)$ are, respectively, $n\times n$, $n\times \ell$ and $m\times n$ time-varying matrices, and $\mathbf{x}(t_o) = \mathbf{x}_o$ is the initial condition. Using the method of variation of parameters, the solution of (1) can be expressed as

$$\mathbf{x}(t) = \mathbf{\Phi}_A(t, t_o)\mathbf{x}(t_o) + \int_{t_o}^{t} \mathbf{\Phi}_A(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau, \tag{2a}$$

$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) \tag{2b}$$

* loh@oakland.edu and cheok@oakland.edu.

where $\mathbf{\Phi}_A(t, t_o)$ denotes the $n\times n$ state transition matrix associated with $\mathbf{A}(t)$, and satisfies

$$\dot{\mathbf{\Phi}}_A(t, t_o) = \mathbf{A}(t)\,\mathbf{\Phi}_A(t, t_o), \quad \mathbf{\Phi}_A(t_o, t_o) = \mathbf{I}_n$$

$\mathbf{I}_n$ denotes the $n\times n$ unit matrix. (3)

The state transition matrix $\mathbf{\Phi}_A(t, \tau)$ is related to the fundamental matrix $\mathbf{X}_A(t)$ by

$$\mathbf{\Phi}_A(t, \tau) = \mathbf{X}_A(t)\mathbf{X}_A^{-1}(\tau), \tag{4}$$

where the fundamental matrix

$$\mathbf{X}_A(t) = \mathrm{e}^{\int_{t_o}^{t} \mathbf{A}(\tau)d\tau} \tag{5}$$

solves the $n\times n$ matrix differential equation

$$\dot{\mathbf{X}}_A(t) = \mathbf{A}(t)\,\mathbf{X}_A(t), \quad \mathbf{X}_A(t_0) = \mathbf{X}_o. \tag{6}$$

Further, the matrix exponential in (5) is defined by the power series $\sum_{k=0}^{\infty} \frac{1}{k!}\left(\int_{t_o}^{t} \mathbf{A}(\tau)d\tau\right)^k$, thereby (5) yields, in general, a solution of the form

$$\mathbf{X}_A(t) = \exp\left(\int_{t_o}^{t} \mathbf{A}(\tau)d\tau\right) = \sum_{k=0}^{\infty} \frac{1}{k!}\left(\int_{t_o}^{t} \mathbf{A}(\tau)d\tau\right)^k. \tag{7}$$

However, if $\left(\int_{t_o}^{t} \mathbf{A}(\tau)d\tau\right)^{m+1} = \mathbf{0}$ for some finite $m < \infty$, then (7) becomes a finite-sum solution given by [18]

$$\mathbf{X}_A(t) = \mathbf{I}_n + \int_{t_o}^{t} \mathbf{A}(\tau)d\tau + \cdots + \frac{1}{m!}\left(\int_{t_o}^{t} \mathbf{A}(\tau)d\tau\right)^m. \tag{8}$$

It is well known that determining the matrix exponential given by (5) is a difficult task, even for constant matrix $\mathbf{A}$ [13]. Note also that $\mathbf{X}_A(t)$ is nonsingular for all t, but may be nonunique; however, the state transition matrix given by $\mathbf{\Phi}_A(t, t_o) = \mathbf{X}_A(t)\mathbf{X}_A^{-1}(t_o)$ is unique for all t_o and t.

2 Facts about LTV Systems

Consider an LTV system described by the homogenous ordinary differential equation (ODE)

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}\,, \quad \mathbf{x}(t_o) = \mathbf{x}_o\,. \tag{9}$$

The following are important facts about the fundamental matrix $\mathbf{X}_A(t)$ and state transition matrix $\mathbf{\Phi}_A(t, t_o)$ associated with $\mathbf{A}(t)$:

F1: Sufficient Conditions for the Existence of Fundamental and State Transition Matrices

The conditions are summarized in Table 1 ([2, 7, 9, 12, 14-16]).

Table 1: Sufficient conditions for the existence of analytical solutions of $\mathbf{X}_A(t)$ and $\mathbf{\Phi}_A(t, t_o)$

(i)	$\mathbf{A}(\tau)$ has piecewise continuous elements $\{a_{ij}(\tau)\}$ for all i, j and $\tau \in [t_o,\ t]$;
(ii)	$\mathbf{A}(t)$ commutes with its integral $\int_{t_o}^{t} \mathbf{A}(\tau)d\tau$, i.e., $\mathbf{A}(t)\left(\int_{t_o}^{t} \mathbf{A}(\tau)d\tau\right) = \left(\int_{t_o}^{t} \mathbf{A}(\tau)d\tau\right)\mathbf{A}(t)$;
(iii)	$\mathbf{A}(t)\mathbf{A}(s)=\mathbf{A}(s)\mathbf{A}(t)$ for all t and s;
(iv)	$\int_{t_o}^{t_1} \mathbf{A}(\tau)d\tau \int_{t_o}^{t_2} \mathbf{A}(s)ds = \int_{t_o}^{t_2} \mathbf{A}(s)ds \int_{t_o}^{t_1} \mathbf{A}(\tau)d\tau$;
(v)	$\mathbf{A}(t) = \alpha(t)\mathbf{A}$, where $\alpha(t)$ is a scalar function and $\mathbf{A}$ is a constant matrix;
(vi)	$\mathbf{A}(t) = \sum_{i=1}^{m} \alpha_i(t)\mathbf{A}_i$, where $\alpha_i(t)$ are scalar functions, and $\mathbf{A}_i$ are constant matrices such that $A_iA_j=A_jA_i$, i.e., A_i and A_j commute for all $\{i, j\}=\{1, 2, \cdots, m\}$;
(vii)	$\mathbf{A}(t)$ can be diagonalized as $\mathbf{D}(t) = \mathbf{T}(t)^{-1}\mathbf{A}(t)\mathbf{T}(t)$, where $\mathbf{D}(t) = diag\left(\lambda_1(t), \ldots, \lambda_n(t)\right)$ and $\{\lambda_1(t), \ldots, \lambda_n(t)\}$ denote the eigenvalues of $\mathbf{A}(t)$. $\mathbf{T}(t)$ is the similarity transformation matrix.

F2: Properties of Diagonal, Upper and Lower Triangular Matrices

A general upper triangular matrix $\mathbf{U}(t)$ and a lower triangular matrix $\mathbf{L}(t)$ have the forms, respectively,

$$\mathbf{U}(t) = \begin{bmatrix} a_{11}(t) & a_{12}(t) & \cdots & a_{1n}(t) \\ 0 & a_{22}(t) & \cdots & a_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn}(t) \end{bmatrix} \text{ and}$$

$$\mathbf{L}(t) = \begin{bmatrix} a_{11}(t) & 0 & \cdots & 0 \\ a_{21}(t) & a_{22}(t) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}(t) & a_{2n}(t) & \cdots & a_{nn}(t) \end{bmatrix} \tag{10}$$

where $\{a_{ij}\}$ are the elements. Their properties help find the corresponding fundamental and state transition matrices $\mathbf{X}(t)$ and $\mathbf{\Phi}(t, 0)$:

Properties of upper diagonal matrix $\mathbf{U}$ and lower diagonal matrix $\mathbf{L}$:

- $\mathbf{U}^T = \mathbf{U}$ and $\mathbf{L}^T = \mathbf{L}$, where $(.)^T$ denotes the transpose of (.);
- The product of two upper triangular or lower triangular matrices is, respectively, an upper triangular or lower triangular matrix, i.e., $\mathbf{U}_1\mathbf{U}_2 = \mathbf{U}$ and $\mathbf{L}_1\mathbf{L}_2 = \mathbf{L}$;
- A diagonal matrix $\mathbf{D}(t)$ is invertible if and only if all its diagonal elements are nonzero;
- Diagonal matrices $\mathbf{D}(t)$ always commute, i.e., $\mathbf{D}(t)\mathbf{D}(s)=\mathbf{D}(s)\mathbf{D}(t)$;
- Upper and lower triangular matrices with identical diagonal elements are commutative; furthermore, if all their diagonal elements are zeros, they become nilpotent matrices. In addition, the eigenvalues of upper and lower triangular matrices are equal to their diagonal elements.
- The eigenvalues of an $n \times n$ time-varying matrix $\mathbf{A}(t)$ can be determined by using the conventional characteristic equation, i.e., $\Delta = \det[\lambda(t)\mathbf{I}_n - \mathbf{A}(t)] = 0$ [15].

F3: Method of Superposition Principle (MSP)

An $n \times n$ analytical solution $\mathbf{X}(t)$ for the homogenous LTV system described by ODE (6) can be constructed by picking n linearly independent initial conditions [1-3]. The method is summarized as follows:

Set n normalized independent initial condition (IC) as:

$$\mathbf{x}^1(t_o) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{x}^2(t_o) = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \quad \mathbf{x}^n(t_o) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}. \tag{11}$$

Let $\{\mathbf{x}^i(t),\ i = 1,\ \cdots,\ n\}$ be the solution of $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}$ based on (10), i.e., $\mathbf{x}^i(t)$ is solved uniquely one column at a time for each initial condition $\mathbf{x}^i(t_o)$. This construction is based on the principle of superposition of linear systems, and is a *method of superposition principle (MSP)* with normalized ICs. The resulting $n \times n$ nonsingular matrix, denoted by $\mathbf{X}_{Normalized}(t)$, is given by

$$\mathbf{X}_{Normalized}(t) = \left[\mathbf{x}^1(t) \mid \cdots \mid \mathbf{x}^n(t)\right]. \tag{12}$$

It follows from (12) that the normalized fundamental matrix $\mathbf{X}_{Normalized}(t)$ satisfies

$$\begin{aligned}\dot{\mathbf{X}}_{Normalized}(t) &= \left[\dot{\mathbf{x}}^1(t) \mid \cdots \mid \dot{\mathbf{x}}^n(t)\right] = \mathbf{A}(t)\left[\mathbf{x}^1(t) \mid \cdots \mid \mathbf{x}^n(t)\right] \\ &= \mathbf{A}(t)\mathbf{X}_{Normalized}(t).\end{aligned} \tag{13}$$

The simple initial condition (10) is a convenient choice to determine $\mathbf{X}_{Normalized}(t)$. Note that it is also possible to choose any set of initial conditions $\left\{\bar{\mathbf{x}}^i(t_o),\ i=1,\ 2,\ \cdots,\ n\right\}$, thereby obtaining a different $\bar{\mathbf{X}}(t)$, as long as $\left\{\bar{\mathbf{x}}^i(t_o),\ i=1,\ 2,\ \cdots,\ n\right\}$ are linearly independent vectors. Therefore, the non-uniqueness of $\mathbf{X}_{Normalized}(t)$ and $\bar{\mathbf{X}}(t)$ provides flexibility for analyzing LTV systems; however, the state transition matrix $\mathbf{\Phi}(t, t_o)$ is unique and is given by (4).

Remark 1: The method of superposition principle (MSP) hinges on $\mathbf{A}(t)$ being an upper or lower triangular matrix [6], so that the analytical solution of each ODE $\dot{x}_i = \sum_{j=1}^{n} a_{ij}(t)x_j$ for all $i = 1, 2, ..., n$ and $j = 1, 2, ..., n$ can be determined successively. The method is particularly attractive for **manual calculations** of $\mathbf{X}(t)$ and $\mathbf{\Phi}(t, t_o)$ for low dimensional systems, such as $n = 2$ and $n = 3$.

F4: Matrix Exponentials and Commutativity of Constant and Time-Varying Matrices

Given two square matrices **A** and **B**, it follows, in general, that [5, 7]

$$e^{\mathbf{A}t}\mathbf{B} \neq \mathbf{B}e^{\mathbf{A}t}, \tag{14a}$$

$$e^{\mathbf{A}t}e^{\mathbf{B}t} \neq e^{\mathbf{B}t}e^{\mathbf{A}t}, \tag{14b}$$

$$e^{\mathbf{A}t}e^{\mathbf{B}t} \neq e^{(\mathbf{A}+\mathbf{B})t}. \tag{14c}$$

Equality will hold if and only if **A** and **B** commute, i.e., **AB=BA**. Further, if **AB**=**BA**, then $e^{\mathbf{A}t}e^{\mathbf{B}t} = e^{\mathbf{B}t}e^{\mathbf{A}t} = e^{\mathbf{A}t+\mathbf{B}t} \Rightarrow e^{\mathbf{A}t} = e^{\mathbf{A}t+\mathbf{B}t-\mathbf{B}t}$. In addition, if **Y** is invertible, then $e^{\mathbf{Y}\mathbf{X}\mathbf{Y}^{-1}} = \mathbf{Y}e^{\mathbf{X}}\mathbf{Y}^{-1}$ [11].

F5: Separation Principle for LTI and LTV Systems

2.1 LTI Systems

Consider the LTI system

$$\dot{\mathbf{x}} = \mathbf{Ax}+\mathbf{Bu},\ \ \mathbf{x}(0)=\mathbf{x}o, \tag{15a}$$

$$\mathbf{y} = \mathbf{Cx}, \tag{15b}$$

where **A**, **B** and **C** are constant matrices of compatible dimensions. It follows that the pair [**A**, **B**] is controllable if and only if the pair [$\mathbf{A}^T$, $\mathbf{B}^T$] is observable, which is the well-known property of duality for control and estimation.

The feedback control **u** is assumed to be given by

$$\mathbf{u} = -\mathbf{K}\hat{\mathbf{x}}, \tag{16}$$

where **K** is the feedback gain matrix and $\hat{\mathbf{x}}$ is an estimate of **x** generated by an observer.

Next, an observer for (17) can be constructed as

$$\begin{aligned}\dot{\hat{\mathbf{x}}} &= \mathbf{A}\hat{\mathbf{x}} + \mathbf{Bu} + \mathbf{H}\left(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}\right) \\ &= (\mathbf{A} - \mathbf{BK} - \mathbf{HC})\,\hat{\mathbf{x}} + \mathbf{Hy},\end{aligned} \tag{17}$$

where $\mathbf{u} = -\mathbf{K}\hat{\mathbf{x}}$ given by (18), and **H** is the observer gain matrix.

Define the estimation error as

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} \quad (\Rightarrow \ \mathbf{x} = \hat{\mathbf{x}} + \mathbf{e}), \tag{18}$$

which yields

$$\dot{\mathbf{e}} = \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}}. \tag{19}$$

Further, (19) can be expressed as ■

$$\begin{aligned}\dot{\mathbf{e}} &= (\mathbf{Ax} + \mathbf{Bu}) - \left[(\mathbf{A} - \mathbf{HC})\,\hat{\mathbf{x}} + \mathbf{Bu} + \mathbf{HCx}\right] \\ &= (\mathbf{A}\text{-}\mathbf{HC})\mathbf{e}.\end{aligned} \tag{20}$$

Also using (18), (20) can be expressed as

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} = (\mathbf{A} - \mathbf{BK})\,\mathbf{x} + \mathbf{BKe}. \tag{21}$$

Combing (20) and (21), we obtain the augmented systems

$$\begin{bmatrix}\dot{\mathbf{x}} \\ \dot{\mathbf{e}}\end{bmatrix} = \begin{bmatrix}\mathbf{A} - \mathbf{BK} & \mathbf{BK} \\ \mathbf{0} & \mathbf{A} - \mathbf{HC}\end{bmatrix}\begin{bmatrix}\mathbf{x} \\ \mathbf{e}\end{bmatrix} \triangleq \mathbf{F}\begin{bmatrix}\mathbf{x} \\ \mathbf{e}\end{bmatrix}, \tag{22}$$

which yields the characteristic equation

$$\Delta = \det(\lambda\mathbf{I} - \mathrm{F}) = \det\left(\lambda\mathbf{I} - (\mathbf{A}\text{-}\mathbf{BK})\right)\det\left(\lambda\mathbf{I} - (\mathbf{A}\text{-}\mathbf{BK})\right), \tag{23}$$

whereby

$$\lambda(\mathbf{A} - \mathbf{BK}) \cup \lambda(\mathbf{A} - \mathbf{HC}), \tag{24}$$

where $\lambda(.)$ denotes the eigenvalues of (.). This property is the well-known *Separation Principle* for LTI systems [4, 11].

Further, (22) can be expressed in the $\{\mathbf{x}, \hat{\mathbf{x}}\}$ coordinates as

$$\begin{bmatrix}\dot{\mathbf{x}}\\ \dot{\hat{\mathbf{x}}}\end{bmatrix}=\begin{bmatrix}\mathbf{A}-\mathbf{BK} & \mathbf{BK}\\ \mathbf{0} & \mathbf{A}-\mathbf{BK}-\mathbf{HC}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{x}}\end{bmatrix}+\begin{bmatrix}\mathbf{0}\\ \mathbf{H}\end{bmatrix}\mathbf{y}, \quad (25)$$

which is the closed-loop feedback control system and observer.

Computer simulation of (25) allows responses $\mathbf{x}(t)$ and $\hat{\mathbf{x}}(t)$ to be compared. If the responses $\mathbf{x}(t)$ and $\hat{\mathbf{x}}(t)$ are not satisfactory, for instance, $\hat{\mathbf{x}}(t)$ does not converge to $\mathbf{x}(t)$ quickly and smoothly, then the gain matrices $\mathbf{K}(t)$ and $\mathbf{H}(t)$ may be redesigned.

2.2 LTV Systems

Consider the LTV System

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u}, \quad \mathbf{x}(0)=\mathbf{x}_o, \quad (26a)$$

$$\mathbf{y} = \mathbf{C}(t)\mathbf{x}, \quad (26b)$$

where $\mathbf{A}(t)$, $\mathbf{B}(t)$ and $\mathbf{C}(t)$ are time-varying matrices of compatible dimensions. Following the formulation in Section I above, we assume that the feedback control $\mathbf{u}(t)$ is given by

$$\mathbf{u} = -\mathbf{K}(t)\hat{\mathbf{x}}, \quad (27)$$

where $\mathbf{K}(t)$ is the time-varying feedback gain matrix and $\hat{\mathbf{x}}$ is an estimate of $\mathbf{x}$ generated by an observer. Recall that, for LTI systems, the pair [**A**, **B**] is controllable if and only if the pair [$\mathbf{A}^T$, $\mathbf{B}^T$] is observable [2, Theorem 6.5, Theorem of Duality]. However, for LTV systems, it follows that **[A**(t), **B**(t)] is controllable if and only if $\left[-\mathbf{A}^T(t),\ \mathbf{B}^T(t)\right]$ is observable (see [2], Problem 6.22 and the Solution Manual for the proof).

An observer for (26) and (27) results in

$$\dot{\hat{\mathbf{x}}} = \boldsymbol{A(t)\hat{x} + B(t)u + H(t)[y - C(t)\hat{x}]}$$

$$= \left[\mathbf{A}(t)-\mathbf{B}(t)\mathbf{K}(t)-\mathbf{H}(t)\mathbf{C}(t)\right]\hat{\mathbf{x}}+\mathbf{H}(t)\mathbf{y}, \quad \hat{\mathbf{x}}(t_o)=\hat{\mathbf{x}}_o. \quad (28)$$

Define the estimation error as

$$\mathbf{e} = \mathbf{x}\text{-}\,\hat{\mathbf{x}}, \quad (29)$$

which yields, with (26) and (28),

$$\dot{\mathbf{e}} = \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}}$$

$$= \left[\mathbf{A}(t)-\mathbf{H}(t)\mathbf{C}((t)\right]e. \quad (30)$$

Combing (26) and (30), we obtain the augmented systems

$$\begin{bmatrix}\dot{\mathbf{x}}\\ \dot{\mathbf{e}}\end{bmatrix}=\begin{bmatrix}\mathbf{A}(t)-\mathbf{B}(t)\mathbf{K}(t) & \mathbf{B}(t)\mathbf{K}(t)\\ \mathbf{0} & \mathbf{A}(t)-\mathbf{H}(t)\mathbf{C}(t)\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{e}\end{bmatrix}$$

$$\triangleq \begin{bmatrix}\boldsymbol{A_{cl}(t)} & \boldsymbol{B(t)K(t)}\\ \mathbf{0} & \boldsymbol{A_o(t)}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{e}\end{bmatrix} \triangleq \boldsymbol{G(t)}\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{e}\end{bmatrix}, \quad (31)$$

The characteristic equation of (31) is given by

$$\Delta = \det(\left(\lambda(t)\mathbf{I}-\mathbf{G}(t)\right) = \det\left(\lambda(t)\mathbf{I}-\mathbf{A}_{cl}(t)\right)\ \det\left(\lambda(t)\mathbf{I}-\mathbf{A}_o(t)\right), \quad (32)$$

and yields the *Separation Principle* for LTV systems.

Equation (31) can also be expressed in the $\{\mathbf{x}, \hat{\mathbf{x}}\}$-coordinates as

$$\begin{bmatrix}\dot{\mathbf{x}}\\ \dot{\hat{\mathbf{x}}}\end{bmatrix}=\begin{bmatrix}\mathbf{A}(t) & -\mathbf{B}(t)\mathbf{K}(t)\\ \mathbf{0} & \mathbf{A}t)-\mathbf{B}(t)\mathbf{K}(t)-\mathbf{H}(t)\mathbf{C}(t)\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{x}}\end{bmatrix}+\begin{bmatrix}\mathbf{0}\\ \mathbf{H}(t)\end{bmatrix}\mathbf{y}. \quad (33)$$

Similar to (25) for LTI systems, (33) is needed for implementing LTV closed-loop feedback control systems and observers. Simulation responses $\mathbf{x}(t)$ and $\hat{\mathbf{x}}(t)$ of (33) can be used to adjust the gain matrices $\mathbf{K}(t)$ and $\mathbf{H}(t)$ to improve the performance of $\mathbf{x}(t)$ and $\hat{\mathbf{x}}(t)$.

A block diagram for the LTV feedback control system and LTV observer is shown in Figure 1.

F6:Controllability and Observability of LTV Systems

Consider the LTV system described by (1), repeated here for ease of reference:

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}+\mathbf{B}(t)\mathbf{u}(t), \quad \mathbf{x}(t_o)=\mathbf{x}_o, \quad t_o \in [a,b], \quad (1a)$$

$$\mathbf{y} = \mathbf{C}(t)\mathbf{x}. \quad (1b)$$

We have the following sufficient conditions for the controllability and observability of LTV systems.

Theorem 1: Controllability of LTV systems (2, Th 6.12; 6, [12], Th 2.5; 16)

Let $\mathbf{A}(t)$ and $\mathbf{B}(t)$ be (n-1) times continuously differentiable. Then the pair [$\mathbf{A}(t)$, $\mathbf{B}(t)$] is controllable at t_o if there exists a finite $t_1 > t_o$ such that

$$\text{rank}[\mathbf{M}(t)] = n, \quad (34)$$

where

$$\mathbf{M}(t) = \left[\mathbf{M}_o(t) \mid \mathbf{M}_1(t) \mid \cdots \mid \mathbf{M}_{n-1}(t)\right], \quad (35a)$$

$$\mathbf{M_o}(t) = \mathbf{B}(t), \quad (35b)$$

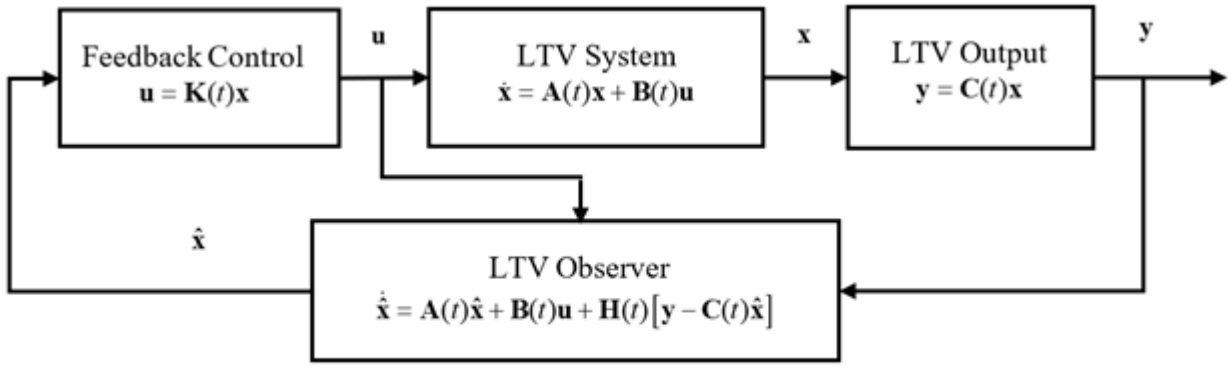


Figure 1. Block diagram for LTV feedback control system with LTV observer

$$\mathbf{M}_{m+1}(t) = -\mathbf{A}(t)\mathbf{M}_m(t) + \dot{\mathbf{M}}_m(t), \tag{35c}$$

for $m = 0, 1, \ldots, n-1$. ■

For LTI systems, (35) yields

$$\mathbf{M} = \left[\mathbf{B} \,\vdots\, -\mathbf{AB} \,\vdots\, \mathbf{A}^2\mathbf{B} \,\vdots\, -\mathbf{A}^3\mathbf{B} \,\vdots\, \cdots \,\vdots\, \mathbf{A}^{n-1}\mathbf{B}\right], \tag{36}$$

which has the same rank as the standard controllability theorem of the pair [A, **B**] given by

$$\bar{\mathbf{M}} = \left[\mathbf{B} \,\vdots\, \mathbf{AB} \,\vdots\, \mathbf{A}^2\mathbf{B} \,\vdots\, \mathbf{A}^3\mathbf{B} \,\vdots\, \cdots \,\vdots\, \mathbf{A}^{n-1}\mathbf{B}\right], \tag{37}$$

i.e., the alternate minus sign in (36) do not affect its rank and $\text{rank}(\bar{\mathbf{M}}) = \text{rank}(\mathbf{M})$.

Theorem 2: Observability of LTV systems [2, Theorem 6.012]

Let $\mathbf{A}(t)$ and $\mathbf{C}(t)$ be $(n-1)$ times continuously differentiable. Then the pair $[\mathbf{A}(t), \mathbf{C}(t)]$ is observable at t_o if there exists a finite $t_1 > t_o$ such that

$$\text{rank}[\mathbf{N}(t)] = n, \tag{38}$$

where

$$\mathbf{N}(t) = \begin{bmatrix} \mathbf{N}_o(t) \\ \mathbf{N}_1(t) \\ \vdots \\ \mathbf{N}_{n-1}(t) \end{bmatrix}, \tag{39a}$$

$$\mathbf{N}_o(t) = \mathbf{C}(t), \tag{39b}$$

$$\mathbf{N}_{m+1}(t) = \mathbf{N}_m(t)\mathbf{A}(t) + \dot{\mathbf{N}}(t), \tag{39c}$$

for $m = 0, 1, \ldots, n-1$. ■

Theorem 3: Relationship between Controllability and Observability of LTV systems [2]

The pair $[\mathbf{A}(t), \mathbf{B}(t)]$ is controllable at t_o if and only if $[-\mathbf{A}^T(t), \mathbf{B}^T(t)]$ is observable at t_o.

Proof: The proof can be found in [2], Solution Manual for Problem 6.22. ■

3 Analytical Solutions and Simulations of LTV Systems, and Design of LTV Feedback Control Systems and LTV Observers

The analytical solutions of the fundamental and state transition matrices of LTV systems and the design of feedback control and observers will be investigated in this section. Matlab solutions and simulations will be given as well.

Example 1: Second-order LTV system [6, 17]

Consider the LTV system:

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} = \begin{bmatrix} -6t^2 & 3t^5 \\ 0 & -3t^2 \end{bmatrix}\mathbf{x}, \quad \mathbf{x}(t_o) = \mathbf{x}_o, \tag{1-1}$$

where $\mathbf{A}(t)$ is an upper triangular matrix. The primary objectives of this example are:

(1a): Solving for $\mathbf{x}(t)$, the fundamental matrix $\mathbf{X}(t)$, and the state transition matrix $\mathbf{\Phi}(t, 0)$;

(1b): Simulating the responses of $\mathbf{x}(t)$ in Matlab with normalized ICs: $x_1(0) = 1$ and $x_2(0) = 1$.

Solution:

(1a): First, we check the matrix commutativity properties of $\mathbf{A}(t)$:

$$\mathbf{A}(t_1)\mathbf{A}(t_2) \neq \mathbf{A}(t_2)\mathbf{A}(t_1) \quad \text{and} \quad \mathbf{A}(t)\mathbf{\Psi}(t) \neq \mathbf{\Psi}(t)\mathbf{A}(t), \tag{1-2}$$

where $\boldsymbol{\Psi}(t)$ denotes the integral of $\mathbf{A}(t)$, i.e.,

$$\boldsymbol{\Psi}(t) = \int_{t_o}^{t} \mathbf{A}(\tau) d\tau \,. \tag{1-3}$$

Since both conditions are not met, then $\mathbf{X}(t) \neq \exp\left(\int_{t_o}^{t} \mathbf{A}(\tau) d \right)$ and $\dot{\mathbf{X}}(t) \neq \mathbf{A}(t)\mathbf{X}(t)$. The problem has been investigated in [6] and [17]. The solutions of Wu and Jain are listed below, respectively:

(i) Wu: $\boldsymbol{\Phi}_{Wu}(t,0) = \left[\begin{array}{c|c} e^{-2t^3} & e^{-2t^3} - e^{-t^3} + t^3 e^{-t^3}) \\ \hline 0 & e^{-t^3} \end{array}\right]$, $\boldsymbol{\Phi}_{Wu}(0,0) = \mathbf{I}_2$, (1-4)

$$\Rightarrow \quad \mathbf{x}(t)_{Wu} = \begin{bmatrix} x_1(t)_{Wu} \\ x_2(t)_{Wu} \end{bmatrix} = \begin{cases} e^{-2t^3} x_1(0) + (e^{-2t^3} - e^{-t^3} + t^3 e^{-t^3}) x_2(0), \\ e^{-t^3} x_2(0). \end{cases} \tag{1-5}$$

(ii) Jain:

$$\boldsymbol{\Phi}_{Jain}(t,0) = \left[\begin{array}{c|c} e^{-2t^3} & \frac{t^3}{2}\left(e^{-t^3} - e^{-2t^3}\right) \\ \hline 0 & e^{-t^3} \end{array}\right], \quad \boldsymbol{\Phi}_{Jain}(0,0) = \mathbf{I}_2, \tag{1-6}$$

$$\Rightarrow \mathbf{x}(t)_{Jain} = \begin{bmatrix} x_1(t)_{Jain} \\ x_2(t)_{Jain} \end{bmatrix} = \begin{cases} e^{-2t^3} x_1(0) + \frac{t^3}{2}\left(e^{-t^3} - e^{-2t^3}\right) x_2(0), \\ e^{-t^3} x_2(0), \end{cases} \tag{1-7}$$

where the state transition matrices $\boldsymbol{\Phi}_{Wu}(t,0) \neq \boldsymbol{\Phi}_{Jain}(t,0)$. It follows that

$$\frac{\partial \boldsymbol{\Phi}_{Wu}(t,0)}{\partial t} = \mathbf{A}(t)\boldsymbol{\Phi}_{Wu}(t,0) \quad \Rightarrow \quad \text{Wu's solution is correct,} \tag{1-8}$$

$$\frac{\partial \boldsymbol{\Phi}_{Jain}(t,0)}{\partial t} \neq \mathbf{A}(t)\boldsymbol{\Phi}_{Jain}(t,0) \quad \Rightarrow \quad \text{Jain's solution is incorrect.} \tag{1-9}$$

If we check more closely, it follows from (1-5) and (1-7) that

$$x_2(t)_{WU} = x_2(t)_{Jain} \,, \tag{1-10}$$

but the error between $x_1(t)_{Wu}$ and $x_1(t)_{Jain}$ is given by

$$e_1(t) = x_1(t)_{Wu} - x_1(t)_{Jain}$$

$$= \left[e^{-t^3}\left(\frac{t^3}{2} - 1\right) + e^{-2t^3}\left(\frac{t^3}{2} + 1\right) \right] x_2(0) \quad \to 0 \quad \text{as } t \to \infty \,. \tag{1-11}$$

Further, the error between $\boldsymbol{\Phi}_{Wu}(t,0)$ and $\boldsymbol{\Phi}_{Jain}(t,0)$ is given by

$$\mathbf{E}(t) = \boldsymbol{\Phi}_{Wu}(t,0) - \boldsymbol{\Phi}_{Jain}(t,0) = \left[\begin{array}{c|c} 0 & e^{-2t^2} - e^{-t^3} + \frac{1}{2}t^3\left(e^{-t^3} + e^{-2t^3}\right) \\ \hline 0 & 0 \end{array}\right]$$

$$\to 0 \quad \text{as } t \to \infty \,. \tag{1-12}$$

The Matlab program using Matlab's **EXPM** command that yields (1-6) is listed below:

```
syms t

A = [-6*t^2   3*t^5; 0   -3*t^2];

M = int(A,t) = [-2*t^3, 1/2*t^6]
               [      0,      -t^3];
```

Xmatlab= simplify(expm(M)) =

```
= [ exp(-2*t^3),   -1/2*t^3*exp(-t^3)*(-1+exp(-t^3))]
  [           0,                          exp(-t^3)],
```

which is identical to (1-6) and is incorrect.

(1b): The responses of the analytical solutions $\mathbf{x}(t)_{Wu}$ and $\mathbf{x}(t)_{Jain}$ given by (1-5) and (1-7), respectively, are plotted in Figure 2. For comparisons, simulations of the time-varying ODE (1-1) using Matlab's ODE45 are also plotted. All the plots in Figure 2 agree with the observations given by (1-10) and (1-11). Hence, based on the matrix differential equations $\frac{\partial \boldsymbol{\Phi}_{Wu}(t,0)}{\partial t} = \mathbf{A}(t)\boldsymbol{\Phi}_{Wu}(t,0)$ and $\frac{\partial \boldsymbol{\Phi}_{Jain}(t,0)}{\partial t} \neq \mathbf{A}(t)\boldsymbol{\Phi}_{Jain}(t,0)$ given by (1-8) and (1-9), the error equation given by (1-11), and all the simulation results, we conclude that Wu's method yields the correct solution to $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}$ and Jain's solutions are incorrect. Recall that Matlab will yield correct solutions if $\mathbf{A}(t)$ given in (1-1) is a commutative matrix.

Example 2: Design of Feedback Control for LTV System and LTV observer with Prescribed Properties

Consider the LTV system described by **[6]**

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)u = \begin{bmatrix} 0 & -1 - \exp(-t) \\ 1 & -\exp(-t) \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \,, \tag{2-1a}$$

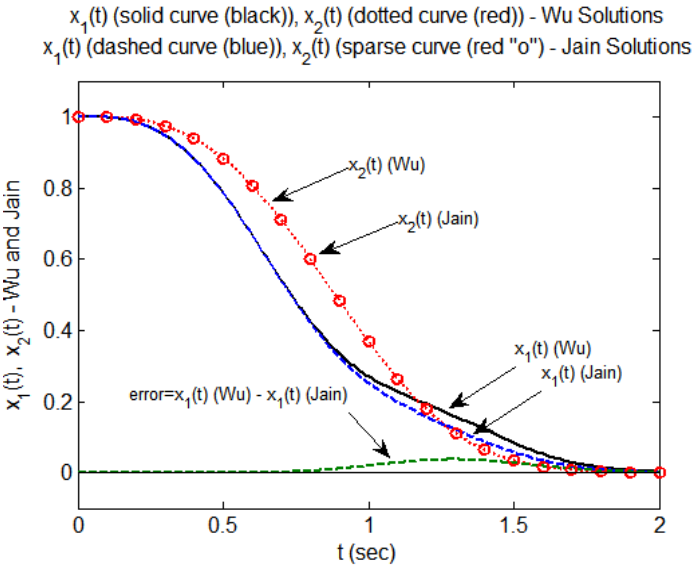


Figure 2: Responses of $\mathbf{x}_{Wu}(t)$ given by (1-4) and $\mathbf{x}_{Jain}(t)$ given by (1-6) with $x_1(0)=1$ and $x_2(0)=1$, and the error $e_1(t)$ given by (1-11). All the solutions decay to zero as $t \to \infty$. ■

$$\mathbf{y}=\mathbf{C}(t)\boldsymbol{x}=[0\ 1]\mathbf{x}=x_2. \tag{2-1b}$$

Do the following:

(2a): Design a feedback control system for (2-1) such that the resulting closed-loop system matrix

$$\mathbf{A}_{cl}(t) = \mathbf{A}(t) - \mathbf{B}(t)\mathbf{K}(t) \tag{2-2}$$

is triangular, commutative, and nilpotent, where $\mathbf{K}(t)$ is the feedback gain matrix. Determine the analytical solutions for the fundamental matrix $\mathbf{X}_{cl}(t)$ and state transition matrix $\mathbf{\Phi}_{cl}(t,0)$ associated with $\mathbf{A}_{cl}(t)$ given by (2-2).

(2b): Design an observer for (2-1) such that the observer matrix

$$\mathbf{A}_o(t) = \mathbf{A}(t) - \mathbf{H}(t)\mathbf{C}(t) \tag{2-3}$$

is triangular, commutative, and nilpotent, where $\mathbf{H}(t)$ is the observer gain matrix. Determine the analytical solutions for the fundamental matrix $\mathbf{X}_o(t)$ and state transition matrix $\mathbf{\Phi}_o(t,0)$ associated with $\mathbf{A}_o(t)$ given by (2-3).

(2c): Simulate and plot the responses of $\mathbf{x}(t)$ and $\hat{\mathbf{x}}(t)$.

Solution:

(2a) Design of LTV Control System

Solution:

Step 1: Check the controllability matrix of the pair [$\mathbf{A}(t)$, $\mathbf{B}(t)$] given by (36), Theorem 1. We obtain:

$$\mathbf{M}(t) = \left[\mathbf{M}_o(t) \,\vdots\, \mathbf{M}_1(t)\right]$$

$$= \left[\mathbf{M}_o(t) \,\vdots\, -\mathbf{A}(t)\mathbf{M}_o(t) + \dot{\mathbf{M}}_o(t)\right] = \left[\mathbf{B}(t) \,\vdots\, -\mathbf{A}(t)\mathbf{B}(t)\right]$$

$$= \begin{bmatrix} 0 & 1+\exp(-t) \\ 1 & \exp(-t) \end{bmatrix}, \tag{2-4a}$$

where

$$\mathbf{M}_o(t) = \mathbf{B}(t), \tag{2-4b}$$

$$\mathbf{M}_1(t) = -\mathrm{A}(t)*\mathbf{Mo}(t) = \begin{bmatrix} 1+\exp(-t) \\ \exp(-t) \end{bmatrix}. \tag{2-4c}$$

The determinant $\det(\mathbf{M}(t))$ of $\mathbf{M}(t)$ is given by

$$\det(\mathbf{M}(t)) = -1 - \exp(-t) \neq 0 \quad \text{for all } t, \tag{2-5}$$

$\Rightarrow$ rank[$\mathbf{M}(t)$] = 2 for all t $\Rightarrow$ LTV system (2.1) is controllable for all t.

Step 2: Design of feedback control system

Let a feedback control be given by

$$u = -\mathbf{K}(t)\,\hat{\mathbf{x}}, \tag{2-6}$$

where $\mathbf{K}(t) = [k_1(t) \quad k_2(t)]$ is a feedback gain matrix, and $\hat{\mathbf{x}}$ is an estimate generated by an observer. Substituting (2-6) into (2-1) yields the closed-loop control system

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} - \mathbf{B}(t)\mathbf{K}(t)\,\hat{\mathbf{x}}. \tag{2-7}$$

Since [$\mathbf{A}(t)$, $\mathbf{B}(t)$] is a controllable pair, a suitable control gain matrix $\mathbf{K}(t)$ exists such that $\mathbf{A}_{cl}(t)$ has desirable properties, specifically, $\mathbf{A}_{cl}(t)$ being triangular and commutative.

Step 3: Determine the control gain matrix $\mathbf{K}(t)$ in (2-6) and (2-7). We have

$$\mathbf{A}_{cl}(t) = \mathbf{A}(t) - \mathbf{B}(t)\mathbf{K}(t) = \begin{bmatrix} 0 & -1-\exp(-t) \\ 1-k_1(t) & -\exp(-t)-k_2(t) \end{bmatrix}. \tag{2-8}$$

Setting $k_1(t) = 1$ and $k_2(t) = -\exp(-t)$ in (2-8) yields

$$\mathbf{A}_{cl}(t)=\mathbf{A}(t)\text{-}\mathbf{B}(t)\mathbf{K}(t)=\begin{bmatrix}0 & -1-\exp(-t)\\ 0 & 0\end{bmatrix}, \qquad (2\text{-}9)$$

which is a commutative and triangular matrix with zero diagonal elements so that the design criteria are satisfied.

Step 4: Determination of the fundamental matrix $\mathbf{X}_{cl}(t)$ and transition matrix $\mathbf{\Phi}_{cl}(t,0)$.

Since $\mathbf{A}_{cl}(t)$ is a triangular and commutative matrix**,** its associated fundamental matrix $\mathbf{X}_{cl}(t)$ and state transition matrix $\mathbf{\Phi}_{cl}(t,0)$ can be determined readily as follows: We have

$$\int_o^t \mathbf{A}_{cl}(\tau)d\tau=\begin{bmatrix}0 & -t+\exp(-t)\\ 0 & 0\end{bmatrix}, \ \left(\int_o^t \mathbf{A}_{cl}(\tau)d\tau\right)^2=\mathbf{0}. \quad (2\text{-}10)$$

Since $\left(\int_o^t \mathbf{A}_{cl}(\tau)d\tau\right)^2=\mathbf{0}$, $\mathbf{X}_{cl}(t)$ is given by

$$\mathbf{X}_{cl}(t)=\mathrm{e}^{\int_o^t \mathbf{A}_{cl}(\tau)d\tau}=\sum_{k=0}^{\infty}\frac{1}{k!}\left(\int_o^t \mathbf{A}_{cl}(\tau)d\tau\right)^k=\mathbf{I}_2+\int_o^t \mathbf{A}_{cl}(\tau)d\tau, \qquad (2\text{-}11)$$

which yields a finite-sum solution

$$\mathbf{X}_{cl}(t)=\mathbf{I}_2+\int_o^t \mathbf{A}_{cl}(\tau)d\tau=\begin{bmatrix}1 & -t+\mathrm{e}^{-t}\\ 0 & 1\end{bmatrix}, \quad \mathbf{X}_{cl}(0)=\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}. \qquad (2\text{-}12)$$

Using (2-12), the state transition matrix $\mathbf{\Phi}_{cl}(t,0)$ is obtained as

$$\mathbf{\Phi}_{cl}(t,0)=\mathbf{X}_{cl}(t)\mathbf{X}_{cl}^{-1}(0)=\begin{bmatrix}1 & -1-t+e^{-t}\\ 0 & 1\end{bmatrix}, \quad \mathbf{\Phi}_{cl}(t,0)=\mathbf{I}_2. \qquad (2\text{-}13)$$

Equations (2-12) and (2-13) satisfy, respectively, the matrix differential equations:

$$\dot{\mathbf{X}}_{cl}(t)=\mathbf{A}_{cl}(t)\mathbf{X}_{cl}(t), \ \mathbf{X}_{cl}(0), \qquad (2\text{-}14)$$

$$\dot{\mathbf{\Phi}}_{cl}(t,0)=\mathbf{A}_{cl}(t)\mathbf{\Phi}_{cl}(t,0), \ \mathbf{\Phi}_{cl}(0,0)=\mathbf{I}_2, \qquad (2\text{-}15)$$

which confirm that $\mathbf{X}_{cl}(t)$ and $\mathbf{\Phi}_{cl}(t,0)$ solve (2-14) and (2-15), respectively.

(2c): Design of LTV Observer with Prescribed Properties

Step 1: Check the observability matrix of the pair [$\mathbf{A}(t)$, $\mathbf{C}(t)$] given by (40), Theorem 2. We obtain:

$$\mathbf{N}(t)=\begin{bmatrix}\mathbf{N}_o(t)\\ \mathbf{N}_1(t)\end{bmatrix}=\begin{bmatrix}0 & 1\\ 1 & -\exp(-t)\end{bmatrix}, \qquad (2\text{-}16)$$

The determinant det($\mathbf{N}(t)$) of $\mathbf{N}(t)$ is given by

$$\det(\mathbf{N}(t)) = -1, \qquad (2\text{-}17)$$

$\Rightarrow$ rank[$\mathbf{N}(t)$] = 2 for all t $\Rightarrow$ LTV system (2.1) is observable for all t.

Step 2: An observer for (2-1) has been designed in [8, Eq. (3.215)] and is given by,

$$\begin{aligned}\dot{\hat{\mathbf{x}}} &= \mathbf{A}(t)\hat{\mathbf{x}}+\mathbf{H}(t)[\mathbf{y}-\mathbf{C}(t)\hat{\mathbf{x}}]+\mathbf{B}(t)\mathbf{u}\\ &= [\boldsymbol{A}(t)-\boldsymbol{H}(t)\boldsymbol{C}(t)]\hat{\boldsymbol{x}}+\boldsymbol{H}(t)\boldsymbol{y}+\boldsymbol{B}(t)\boldsymbol{u}\\ &\triangleq \boldsymbol{A}_o(t)\hat{\boldsymbol{x}}+\boldsymbol{H}(t)\boldsymbol{y}+\boldsymbol{B}(t)\boldsymbol{u}, \end{aligned} \qquad (2\text{-}18)$$

where the observer gain matrix $\mathbf{H}(t)$ is given by

$$\mathbf{H}(t)=\begin{bmatrix}h_1(t)\\ h_2(t)\end{bmatrix}. \qquad (2\text{-}19)$$

Substituting (2-19) into (2-18) yields

$$\dot{\hat{\mathbf{x}}}=\begin{bmatrix}0 & -1-e^{-t}-h_1(t)\\ 1 & -e^{-t}-h_2(t)\end{bmatrix}\hat{\mathbf{x}}+\mathbf{H}(t)\mathbf{y}+\mathbf{B}(t)\mathbf{u}. \qquad (2\text{-}20)$$

In [6], the observer gain matrix $\mathbf{H}(t)$ was chosen as

$$\mathbf{H}(t)=\begin{bmatrix}h_1(t)\\ h_2(t)\end{bmatrix}=\begin{bmatrix}m_o-1-\mathrm{e}^{-t}\\ m_1-\mathrm{e}^{-t}\end{bmatrix}, \qquad (2\text{-}21)$$

where m_o and m_1 are constants. Substituting (2-21) into (2-20) yields an LTI observer

$$\dot{\hat{\mathbf{x}}}=\begin{bmatrix}0 & -m_o\\ 1 & -m_1\end{bmatrix}\hat{\mathbf{x}}+\mathbf{H}(t)\mathbf{y}+\mathbf{B}(t)\mathbf{u} \ \square \ \bar{\mathbf{A}}_o\hat{\mathbf{x}}+\mathbf{H}(t)\mathbf{y}+\mathbf{B}(t)\mathbf{u}. \qquad (2\text{-}22)$$

where $\bar{\mathbf{A}}_o$ is a constant matrix. The eigenvalues of $\bar{\mathbf{A}}_o$ are given by $\lambda=\dfrac{-m_1\pm\sqrt{m^2+4m_o}}{2}$ which can be used to guide the choice of m_o and m_1 for stability analysis.

Simulation studies of the LTV closed-loop control and observer can be obtained by using (2-1) and (2-22), repeated for ease of reference,

$$\dot{\mathbf{x}}=\mathbf{A}(t)\mathbf{x}+\mathbf{B}(t)u=\begin{bmatrix}0 & -1-\exp(-t)\\ 1 & -\exp(-t)\end{bmatrix}\mathbf{x}+\begin{bmatrix}0\\ 1\end{bmatrix}u,$$

$$\mathbf{y}=\mathbf{C}(t)\boldsymbol{x}=[0\ 1]\mathbf{x}=x_2, \quad \mathbf{x}(0))=\mathbf{x}_o; \tag{2-1}$$

$$\dot{\hat{\mathbf{x}}}=\begin{bmatrix}0 & -m_o\\ 1 & -m_1\end{bmatrix}\hat{\mathbf{x}}+\mathbf{H}(t)\mathbf{y}+\mathbf{B}(t)\mathbf{u}\,, \quad \hat{\mathbf{x}}(0)=\hat{\mathbf{x}}_o \tag{2-22}$$

where

$$\underline{u}=-\mathbf{K}(t)\,\hat{\mathbf{x}}=\begin{bmatrix}1 & -\exp(-t)\end{bmatrix}\hat{\mathbf{x}}\,. \tag{2-23}$$

The responses of $\mathbf{x}(t)$ and $\hat{\mathbf{x}}(t)$ are as shown in Figure 3, where $\mathbf{x}(t)$ converges to $\hat{\mathbf{x}}(t)$ quickly and smoothly.

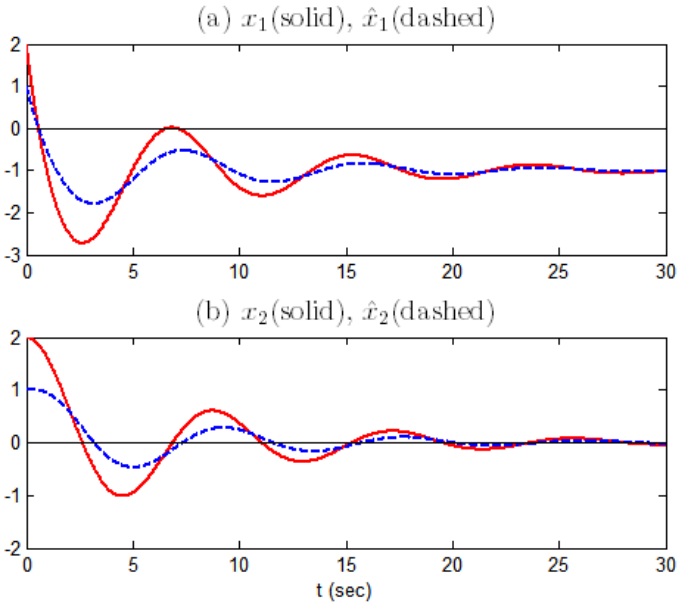


Figure 3: Responses of the estimate $\hat{\mathbf{x}}(t)$ given by (2-22) converges to $\mathbf{x}(t)$ given by (2-1) quickly and smoothly for m_o=1 and m_1=1. The initial conditions were both chosen as $\mathbf{x}(0))=\hat{\mathbf{x}}(0)=[2\ \ 1]^{\mathrm{T}}$ ■

Example 3: Design of Feedback Control for LTV System and LTV observer with prescribed Properties

Consider a 4th-order LTV system described by the homogenous equation

$$\dot{\mathbf{x}}=\mathbf{A}(t)\mathbf{x}=\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 2t^3 & 3t\\ 1 & 0 & 0 & 0\\ 3t^2 & 0 & 0 & 0\end{bmatrix}\mathbf{x}\,. \tag{3-1}$$

It is an LTV system investigated in [10] that has interesting properties, for example,

$$\mathbf{A}(t)\mathbf{A}(s)\ \neq\ \mathbf{A}(s)\mathbf{A}(t), \tag{3-2a}$$

but

$$\mathbf{A}(t)\int_o^t\mathbf{A}(\tau)d\tau=\int_o^t\mathbf{A}(\tau)d\tau\mathbf{A}(t) \quad\Rightarrow\quad \left[\mathbf{A}(t),\ \int_o^t\mathbf{A}(\tau)d\tau\right]=\mathbf{0}. \tag{3-2b}$$

(3a): Design a feedback control system for (3-1) such that the resulting closed-loop system matrix

$$\mathbf{A}_{cl}(t)=\mathbf{A}(t)-\mathbf{B}(t)\mathbf{K}(t) \tag{3-3}$$

is triangular and commutative, where $\mathbf{K}(t)=\begin{bmatrix}k_1 & k_2 & k_3 & k_4\end{bmatrix}$, is the feedback gain matrix. Determine $\mathbf{K}(t)$ and the analytical solutions for the fundamental matrix $\mathbf{X}_{cl}(t)$ and state transition matrix $\mathbf{\Phi}_{cl}(t,0)$ associated with $\mathbf{A}_{cl}(t)$ given by (3-3).

(3b): Design an observer for (3-1) such that the observer matrix

$$\mathbf{A}_o(t)=\mathbf{A}(t)-\mathbf{H}(t)\mathbf{C}(t) \tag{3-4}$$

is triangular and commutative, where $\mathbf{H}(t)=\begin{bmatrix}h_1 & h_2 & h_3 & h_4\end{bmatrix}^T$ is the observer gain matrix. Determine $\mathbf{H}(t)$ and the analytical solutions for the fundamental matrix $\mathbf{X}_o(t)$ and state transition matrix $\mathbf{\Phi}_o(t,0)$ associated with $\mathbf{A}_o(t)$ given by (3-4).

Solutions:

(3a): Design of LTV feedback control system with prescribed properties

Since (3-1) is a system with no input, we need to modify it for feedback control analysis as

$$\dot{\mathbf{x}}=\mathbf{A}(t)\mathbf{x}+\mathbf{B}(t)\mathbf{u}, \tag{3-5}$$

where $\mathbf{B}(t)$ is to be chosen such that the rows of $\mathrm{B}(t)\mathbf{K}(t)$ can modify the rows of $\mathbf{A}(t)$ to mach the design criteria, in particular $\mathbf{A}_{cl}(t)$ being a triangular and commutative matrix.

Design algorithm:

Step 1: Examining the structure of $\mathbf{A}(t)$, it follows that an upper triangular and commutative matrix $\mathbf{A}_{cl}(t)$ can be obtained by replacing the terms $2t^3$ and $3t$ in the second row of $\mathbf{A}(t)$ by an 0 (zero). This suggests that $\mathbf{B}(t)$ should be chosen with a nonzero row and has the form

$$\mathbf{B}(t)=\begin{bmatrix}0\\ 1\\ 0\\ 0\end{bmatrix}, \tag{3-6}$$

Substituting (3-6) into (3-3) yields

$$\mathbf{A}_{cl}(t)=\mathbf{A}(t)-\mathbf{B}(t)\mathbf{K}(t)=\begin{bmatrix}0 & 0 & 0 & 0\\ -k_1 & -k_2 & 2t^3-k_3 & 3t-k_4\\ 1 & 0 & 0 & 0\\ 3*t^2 & 0 & 0 & 0\end{bmatrix}. \tag{3-7}$$

Setting k_1=0, k_2=0, k_3= $2t^3$ and k_4=3t into (3-7) yields

$$\mathbf{A}_{cl}(t)=\mathbf{A}(t)-\mathbf{B}(t)\mathbf{K}(t)=\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0\\ 3*t^2 & 0 & 0 & 0\end{bmatrix}, \tag{3-8}$$

which is a lower triangular and commutative matrix. It is emphasize that $\mathbf{A}_{cl}(t)$ given by (3-8) was obtained without regard to whether the pair [$\mathbf{A}(t)$, $\mathbf{B}(t$] is controllable or uncontrollable.

Step 2: Check the controllability matrix $\mathbf{M}(t)$ of the pair [$\mathbf{A}(t)$, $\mathbf{B}(t)$] given by (36), Theorem 1 with $\mathbf{B}(t)$ given by (3-6). We have:

$$\mathbf{M}(t)=\begin{bmatrix}\mathbf{M}_o(t) & \mathbf{M}_1(t) & \mathbf{M}_2(t) & \mathbf{M}_3(t)\end{bmatrix}, \tag{3-9a}$$

where

$$\boldsymbol{M}_o(t)=\mathbf{B}(t), \tag{3-9b}$$

$$\boldsymbol{M}_1(t)=-\mathbf{A}(t)\mathbf{M}_o(t)+\dot{\mathbf{M}}_o(t)=\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\quad \mathbf{M}_2(t)=\begin{bmatrix}0\\0\\0\\0\end{bmatrix},$$

$$M_3(t)=-\mathbf{A}(t)\mathbf{M}_2(t)+\dot{\mathbf{M}}_2(t)=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}. \tag{3-9c}$$

which yields

$$\mathbf{M}(t)=\begin{bmatrix}0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}. \tag{3-9d}$$

The determinant det($\mathbf{M}(t)$) of $\mathbf{M}(t)$) is given by

$$\det(\mathbf{M}(t))=0 \quad \Rightarrow \quad \text{LTV system is uncontrollable.} \tag{3-10}$$

Hence, there exists no feedback gain matrix $\mathbf{K}(t)$ in (3-3) that can change the structure of $\mathbf{A}(t)$ to that of $\mathbf{A}_{cl}(t)$. However, the method proposed in the paper has just accomplished the design objective as shown by (3-8). On the other hand, given a general $\mathbf{A}(t)$, the proposed method may fail.

Step 3: Determination of the fundamental matrix $\mathbf{X}_{cl}(t)$ and transition matrix $\boldsymbol{\Phi}_{cl}(t,0)$.

Since $\mathbf{A}_{cl}(t)$ given by (3-8) is a triangular and commutative matrix**,** its associated fundamental matrix $\mathbf{X}_{cl}(t)$ and state transition matrix $\boldsymbol{\Phi}_{cl}(t,0)$ can be determined readily as follows. We have

$$\int_o^t \mathbf{A}_{cl}(\tau)d\tau=\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ t & 0 & 0 & 0\\ t^2 & 0 & 0 & 0\end{bmatrix} \text{ and } \left(\int_o^t \mathbf{A}_{cl}(\tau)d\tau\right)^2=\mathbf{0}. \tag{3-11}$$

Since $\left(\int_o^t \mathbf{A}_{cl}(\tau)d\tau\right)^2=\mathbf{0}$, $\mathbf{X}_{cl}(t)$ is given by

$$\mathbf{X}_{cl}(t)=\mathrm{e}^{\int_o^t \mathbf{A}_{cl}(\tau)d\tau}=\sum_{k=0}^{\infty}\frac{1}{k!}\left(\int_o^t \mathbf{A}_{cl}(\tau)d\tau\right)^k=\mathbf{I}_4+\int_o^t \mathbf{A}_{cl}(\tau)d\tau, \tag{3-12}$$

which yields the finite-sum solution

$$\mathbf{X}_{cl}(t)=\mathbf{I}_4+\int_o^t \mathbf{A}_{cl}(\tau)d\tau=\mathrm{X}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ t & 0 & 1 & 0\\ t^3 & 0 & 0 & 1\end{bmatrix},\quad \mathbf{X}_{cl}(0)=\mathbf{I}_4. \tag{3-13}$$

Using (3-13), the state transition matrix $\boldsymbol{\Phi}_{cl}(t,0)$ is obtained as

$$\boldsymbol{\Phi}_{cl}(t,0)=\mathbf{X}_{cl}(t)\mathbf{X}_{cl}^{-1}(0)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ t & 0 & 1 & 0\\ t^3 & 0 & 0 & 1\end{bmatrix},\quad \boldsymbol{\Phi}_{cl}(t,0)=\mathbf{I}_4. \tag{3-14}$$

Equations (3-13) and (3-14) satisfy, respectively, the matrix differential equations:

$$\dot{\mathbf{X}}_{cl}(t)=\mathbf{A}_{cl}(t)\mathbf{X}_{cl}(t),\ \ \mathbf{X}_{cl}(0), \tag{3-15}$$

$$\dot{\boldsymbol{\Phi}}_{cl}(t,0)=\mathbf{A}_{cl}(t)\boldsymbol{\Phi}_{cl}(t,0),\quad \boldsymbol{\Phi}_{cl}(0,0)=\mathbf{I}_4, \tag{3-16}$$

which confirm that $\mathbf{X}_{cl}(t)$ and $\boldsymbol{\Phi}_{cl}(t,0)$ solve (3-15) and (3-16), respectively.

(3b): Design of LTV Observer with Prescribed Properties

Since (3-1) is a system with no output, we need to modify it for observer design as

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}, \quad \mathbf{y} = \mathbf{C}(t)\mathbf{x}, \tag{3-17}$$

where $\mathbf{C}(t)$ is to be chosen such that $\mathbf{H}(t)\mathbf{C}(t)$ satisfies the design criteria. An LTV observer for (3-17) can be designed as

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}(t)\hat{\mathbf{x}} + \mathbf{H}(t)[\mathbf{y} - \mathbf{H}(t)C(t)\hat{\mathbf{x}}]$$

$$= [\mathbf{A}(t) - \mathbf{H}(t)\mathbf{C}(t)]\hat{\mathbf{x}} + \mathbf{H}(t)\mathbf{y} = \mathbf{A}_o(t)\hat{\mathbf{x}} + \mathbf{H}(t)\mathbf{y}, \tag{3-18}$$

where $\mathbf{H}(t)$ is the observer gain matrix given by

$$\mathbf{H}(t) = \begin{bmatrix} h_1(t) \\ h_2(t) \\ h_3(t) \\ h_4(t) \end{bmatrix}. \tag{3-19}$$

Design Algorithm

Step 1: Once again, examining the structure of $\mathbf{A}(t)$, it follows that a lower triangular and commutative matrix $\mathbf{A}_o(t)$ can be obtained by replacing the third and fourth elements in the first column of $\mathbf{A}(t)$. This suggests that $\mathbf{C}(t)$ can be chosen as

$$\mathbf{C}(t)=[1 \quad 0 \quad 0 \quad 0]. \tag{3-20}$$

Substituting (3-20) into (3-18) yields

$$\mathbf{A}_o(t)=\begin{bmatrix} -h_1 & 0 & 0 & 0 \\ -h_2 & 0 & 2t^3 & 3t \\ 1-h_3 & 0 & 0 & 0 \\ 3t^2 - h_4(t) & 0 & 0 & 0 \end{bmatrix}. \tag{3-21}$$

Setting h_1=0, h_2=0, h_3=1 and h_4= $3t^2$ yields

$$\mathbf{A}_o(t)=\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 2t^3 & 3t \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3-22}$$

which is an upper triangular and commutative matrix.

Step 2: Check the observability matrix N(t) of the pair [$\mathbf{A}(t)$, $\mathbf{C}(t)$] given by (40), Theorem 2, for n=4. We have:

$$\mathbf{N}(t) = \begin{bmatrix} \mathbf{N}_o(t) & \mathbf{N}_1(t) & \mathbf{N}_2(t) & N_3(t) \end{bmatrix}, \tag{3-23a}$$

where

$$\mathbf{N}_o(t)=\mathbf{C}(t), \tag{3-23b}$$

$$N_1(t)= \mathbf{N}_o(t)\mathbf{A}(t) + \dot{\mathbf{N}}_o(t) = \mathbf{0}, \; \mathbf{N}_2(t) = \mathbf{N}_1(t)\mathbf{A}(t) + \dot{\mathbf{N}}_1(t) = \mathbf{0},$$

$$\mathbf{N}_3(t) = \mathbf{N}_2\mathbf{A}(t) + \dot{\mathbf{N}}_2(t) = \mathbf{0}, \tag{3-23c}$$

which yields

$$\mathbf{N}(t)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3-23d}$$

The determinant det(**N**(t)) of **N**(t)) is given by

$$\det(\mathbf{N}(t))=0 \quad \Rightarrow \quad \text{LTV system is uncontrollable.} \tag{3-24}$$

Hence, there exists no observer gain $\mathbf{H}(t)$ that can change the structure of A(t) to that of $\mathbf{A}_{cl}$(t). But the method proposed in this paper has just accomplished the design objective as shown by (3-22). On the other hand, given a general $\mathbf{A}(t)$, the proposed method may not work.

Step 3: Determination of the fundamental matrix $\mathbf{X}_o(t)$ and transition matrix $\mathbf{\Phi}_o(t,0)$.

Since $\mathbf{A}_o(t)$ given by (3-22) is a triangular and commutative matrix**,** its associated fundamental matrix $\mathbf{X}_o(t)$ and state transition matrix $\mathbf{\Phi}_o(t,0)$ can be determined readily as follows: We have

$$\int_o^t \mathbf{A}_o(\tau)d\tau = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5t^4 & 1.5t^2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \left(\int_o^t \mathbf{A}_o(\tau)d\tau\right)^2 = \mathbf{0}. \tag{3-25}$$

Since $\left(\int_o^t \mathbf{A}_o(\tau)d\tau\right)^2 = \mathbf{0}$, $\mathbf{X}_o(t)$ is given by

$$\mathbf{X}_o(t) = \mathrm{e}^{\int_o^t \mathbf{A}_o(\tau)d\tau} = \sum_{k=0}^{\infty} \frac{1}{k!}\left(\int_o^t \mathbf{A}_o(\tau)d\tau\right)^k = \mathbf{I}_4 + \int_o^t \mathbf{A}_o(\tau)d\tau, \tag{3-26}$$

which yields a finite-sum solution

$$\mathbf{X}_{cl}(t) = \mathbf{I}_4 + \int_o^t \mathbf{A}_{cl}(\tau)d\tau = \mathrm{X} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0.5t^4 & 1.5t^2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{X}_{cl}(0) = \mathbf{I}_4. \tag{3-27}$$

Using (3-26), the state transition matrix $\mathbf{\Phi}_o(t,0)$ is obtained as

$$\mathbf{\Phi}_o(t,0)=\mathbf{X}_o(t)\mathbf{X}_o^{-1}(0)=\begin{bmatrix}1&0&0&0\\0&1&0.5t^4&1.5t^2\\0&0&1&0\\0&0&0&1\end{bmatrix},\ \mathbf{\Phi}_o(t,0)=\mathbf{I}_4\,. \tag{3-28}$$

Equations (3-27) and (3-14) satisfy, respectively, the matrix differential equations:

$$\dot{\mathbf{X}}_o(t)=\mathbf{A}_o(t)\mathbf{X}_o(t),\quad \mathbf{X}_o(0)\,, \tag{3-29}$$

$$\dot{\mathbf{\Phi}}_o(t,0)=\mathbf{A}_o(t)\mathbf{\Phi}_o(t,0),\quad \mathbf{\Phi}_o(0,0)=\mathbf{I}_4\,, \tag{3-30}$$

which confirm that $\mathbf{X}_0(t)$ and $\mathbf{\Phi}_o(t,0)$ solve (3-29) and (3-30), respectively.

4 Conclusions

Determination of the analytical solutions for the fundamental and state transition matrices associated with linear time-varying (LTV) systems of the form given by (5), which is a matrix exponential, was investigated. It is well known that determining a matrix exponential is a difficult task. The investigation consisted of two types of LTV systems, namely, feedback control systems and observers. For the design of LTV control systems, one of the main objectives was to require the closed-loop system matrices to have specific structures, specifically, being upper or lower triangular matrices with identical elements on their main diagonal. The same objective was imposed on the observer matrices. Upper and lower triangular matrices have many desirable properties, such as they are commutative as wasstated in fact F.2 in the paper. The commutativity of a matrix will ease the determination of its fundamental and state transition matrices. Examples were given to demonstrate the analysis and design. Simulations and Matlab solutions using its command **EXPM** were provided as well. Future research will address disturbance cancellation control of LTV systems and the design of unknown input LTV observers.

References

[1] B. Bamieh, "Lecture 5: Continuous-Time Linear State-Space Models," University of California, Fall 1999.

[2] C. T. Chen, *Linear Systems and Design*, 4th Ed., New York: Oxford University Press, 2013.

[3] Mohammed Dahleh, Munther Dahleh, and G. Verghese, "Chapter 11, Lectures on Dynamic Systems and Control," MIT6_241JS11, https://www.studocu.com/en/us/document/university-of-massachusetts-amherst/feedback-control-systems/mit6-241js11-chap26-handouts/8198958, 2013/2014.

[4] F. G. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, Fourth Edition, New Jersey: Prentice Hall, 2002.

[5] Grant B. Gustafson, "Systems of Differential Equations, Chapter 11," pp. 740-821, https://www.math.utah.edu/~gustafso/2250systems-de.pdf, Jan 17, 2017.

[6] Vanita Jain and B. K. Lande, "Computation of the Transition Matrix for General Linear-Varying Systems," *International Journal of Engineering & Technology (IJERT)*, 1(6):1-10, August 2012.

[7] T. Kailath, *Linear Systems*, New Jersey: Prentice-Hall, 1980.

[8] Edward W. Kamen, "Fundamentals of Linear Time-Varying Systems," *The Control Systems Handbook*, 2nd Edition, Chapter 3, pp. 3-1 to 3-33, December 2010.

[9] T. Kamizawa, "On Functionally Commutative Quantum Systems," Faculty of Physics, Astronomy and Informatics, Nicolaus Copernicus University, Toru'n, Poland, 2018.

[10] J. F. P. Martin, "Some Results on Matrices which Commute with Their Derivatives," SIAM J. Affl. Math, 15(8):1171-1183, September 1967.

[11] Matrix Calculus, Wikipedia. , https://en.wikipedia.org/wiki/Matarix_calculus, Revised May 2022

[12] Thomas Meurer, "Chapter 2, Analysis and Control of Linear Time-Varying Systems," https://www.control. tf. uni-kiel.de/en/teaching/summer-term/nonlinear-control-systems/nonlinear-control-systems-etit-5013-01.

[13] Cleve Moler and Charles Van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later," *SIAM Review, Society for Industrial and Applied Mathematics,* 45(1):3-49, March 2003.

[14] A. F. Taha, "Computation of State Transition Matrix, Module 04 - Linear Time-Varying Systems," https://ceid.utsa.edu/ataha/wp-content/upload/sites/38/2017/07/EE5143_Module4.pdf, September 26, 2017.

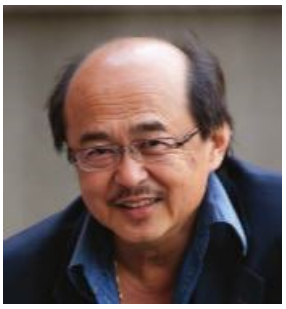
[15] J. M. Wang, "Explicit Solution and Stability of Linear Time-varying Differential State Space Systems," International Journal of Control, Automation and Systems 15(4):1553-1560, 2017.

[16] D. M. Wiberg, *Theory and Problems of State Space and Linear Systems*, New York, McGraw-Hill, Inc., 1971.

[17] M. Y. Wu, "A New Method of Computing the State Transition Matrix of Linear Time-Varying Systems," *Proceedings of the IEEE International Symposium on Circuits and Systems*, San Francisco, pp. 269-272, 1974.

[18] J. Zhu and C. H. Morales, "On Linear Ordinary Differential Equations with Functionally Commutative Coefficient Matrices," *Linear Algebra and Its Applications*, 170:81-105, 1992.

Robert N. K. Loh received his PhD degree in Electrical Engineering from the University of Waterloo, Canada, in 1968. He has taught at various universities since graduation, and has won three endowed chair professorships on two continents - North America and Asia. He was a Senior Vice President of Engineering of an international corporation in Hong Kong, and was an editor and associated editor of several international technical journals. He retired in 2016, but is still very active in his research in control engineering, estimation theory (including stochastic processes), and systems engineering.



Ka C. Cheok is a Professor of Engineering and John Dodge Chair at the Department of Electrical & Computer Engineering, Oakland University, Rochester, MI. He has completed several successful R&D collaborations in intelligent systems and autonomous robotics for over the years. They include fuzzy logic-based highway and city street lane centering systems; ultra-wideband tracking of omnidirectional mobile robots & assets in GPS denied areas; mine-detection robot that sweeps for anti-personnel ordinance, and automated breast cancer diagnostic tester that integrates IR thermography and AI. He has published over 170 technical journal and conference papers, and nine US patents. Dr. Cheok is a co-founder and co-organizer of the annual Intelligent Ground Vehicle Competition, since1993. He served a Consultant Member on the prestigious US Army Science Board.

Logical Modeling of Adiabatic Logic Circuits using VHDL with Examples

Lee A. Belfore II *
Old Dominion University, Norfolk, Virginia, 23529, USA

Abstract

The underlying nature of adiabatic circuits is most accurately characterized at the circuit level as it is for traditional technologies. However, in order to scale system designs for adiabatic logic technologies, modeling of adiabatic circuits at the logic level is necessary. Logic level models of adiabatic logic circuits can facilitate the design, development, and verification of large scale digital systems that may be infeasible using circuit simulators. Adiabatic logic circuits can be powered with a four stage power clock consisting of idle, charge, hold, and recover stages that provides for adiabatic charging and charge recovery to give adiabatic circuits their low power operation. By both discretizing the temporal aspects of the power clock and the logic values, a logical model of adiabatic circuit operation is proposed. Using the expressive capabilities of Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL), the salient aspects of adiabatic circuit models can be captured. In this work, a VHDL framework is defined for modeling adiabatic logic circuits & systems and its use is demonstrated in several example adiabatic logic circuits.

Key Words: Low power electronics; Digital circuits; Logical Model; Digital simulation; VHDL.

1 Introduction

Adiabatic logic circuit technology offers lower power consumption compared with CMOS technologies by energizing circuits adiabatically and then adiabatically recovering stored energy from the circuit for later reuse [3, 7, 8, 10]. The efficiency and behavior is established by the circuit level behaviors that are quantified in circuit simulations and measured in actual circuits. Once the circuits are suitably characterized, the overall operation can be described symbolically. This description of their operation is the basis for logical models of adiabatic circuits.

With current digital system design requirements and modeling practices, it is impractical to rely solely on circuit simulations to validate a design because the circuit simulations require substantial computational resources. As a result, the design process employs conservative models with conservative margins, substitutes vetted approximate models for high fidelity models, and/or, limits circuit simulation to special cases requiring circuit level fidelity. The fidelity is not reduced in an arbitrary fashion, but rather aspects of the circuit operation are modeled symbolically. Circuit level properties associated with the symbolic representations can be included using a circuit extraction step to improve the fidelity of the model. Such is the case with circuit delays, for example.

Approaches for modeling adiabatic and partially adiabatic circuits appear in the literature [14, 15]. In Varga et al., the adiabatic pipeline is modeled using the IEEE `std_logic` type for logic values and guarded blocks to manifest the timing of the power clock [14]. The motivation is to model the pipeline structure in anticipation of synthesis. Finally, approaches for modeling in Verilog are developed [15] with the observation that VHDL is similarly capable. The clear intent of these approaches is to facilitate modeling larger scale models and support synthesis based on the logical behavior of the models.

More generally in the literature, adiabatic circuit dynamics can be modeled with VHDL by one of two methods. First, VHDL libraries can be created with the specific capacity to model analog signals [9]. In addition, the VHDL standard has been extended to support mixed analog/digital modeling in VHDL-AMS [1]. In both of these approaches, the circuit is ultimately represented by a system of differential equations. In these works, it would be necessary to develop libraries to support adiabatic circuit models. The principle disadvantage in these approaches is the significant simulation time necessary for large circuits. During system development, it is more pragmatic to focus on logical modeling, constrained by conservative performance metrics, to facilitate iterative design. Once the design approaches the final phase, then it may become necessary to shift to higher fidelity circuit simulations.

In this work, an approach is introduced for modeling adiabatic circuits. The model defines a multivalued logic value definition consistent with adiabatic circuit operating modes. The logic values facilitate developing adiabatic logic pipelines and troubleshooting of logic circuits. Importantly, the model preserves the dual rail nature of adiabatic signals.

This paper is organized into five sections including an introduction, an overview of the operation of adiabatic circuits, a presentation of adiabatic VHDL models, simulation results for

*Department of Electrical and Computer Engineering. Email: LBelfore@odu.edu

several examples, and a summary.

2 Adiabatic Logic Circuits Operation

In this section, the basis for logical models of adiabatic circuit operation is presented. The intention is to identify modes of circuit operation that can be represented symbolically rather than actual circuit level behaviors. The interested reader can find the details of adiabatic circuit operation elsewhere [2, 3, 7, 8, 10].

Adiabatic circuits are capable of low power operation by providing the energy to the circuit adiabatically and then later adiabatically retrieving the energy for subsequent reuse. Note that adiabatic operation implies that no heat is dissipated during circuit operation. Since the circuit operation is not ideally adiabatic, some energy will be dissipated, but can be greatly reduced compared to traditional CMOS technologies.

2.1 Power Clocks

Adiabatic circuit operation can be divided into four segments reflecting the modes of circuit operation. The segments are *idle*, *charge*, *hold*, & *recover*, or abbreviated by I, C, H, & R. The nature of each segment captures an adiabatic circuit's mode of operation and is manifest by the nature of the power source during the segment as a function of time. Repeating the segments in the order presented enables adiabatic operation. Since segments are repeated periodically, the circuit's power source is henceforth termed the *power clock*.

In more detail, the segment operation is described as follows. In the *idle* mode, the circuit voltage source is 0V, the circuit is unpowered, and thus consumes no power. In the *charge* mode, the voltage supplied slowly increases, charging capacitive elements in the circuit that when fully charged, enabling the circuit to provide its designated function. A key aspect of the charge mode is the "slow" increase in the voltage supplied. By "trickle charging" the capacitive elements, the net power consumed can be shown to be reduced [7]. In the limit where the voltage increases over an indefinitely long time interval, the circuit operates in a truly adiabatic fashion. In the *hold* mode, the circuit is fully charged. With no current entering the circuit, the circuit consumes no power. Finally, in the *recover* mode, the circuit is discharged through its voltage source, returning its charge for later reuse. Similar to the charge mode, the slow ramp down of the voltage source retrieves the charge adiabatically. An important observation is that, unlike traditional CMOS circuits where such charge is resistively dissipated, the charge is recovered through the voltage source for later reuse.

Further, to simplify the discussions, a trapezoidal clock is assumed, although many adiabatic circuits operate using sinusoidal or other periodic shapes that are more easily generated. Figure 1 shows four power clock phases, shifted 90° with respect to one another.

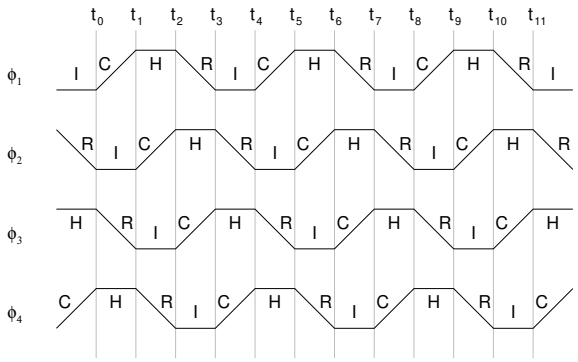
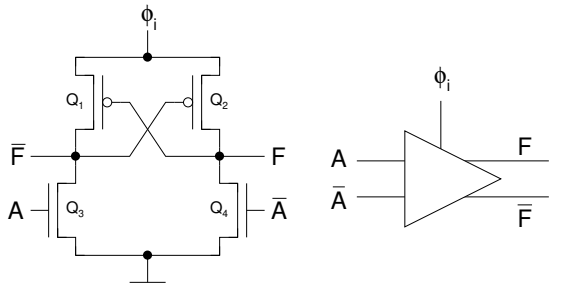


Figure 1: Four power clock phases

2.2 Adiabatic Circuit Dynamic Behavior

Figure 2 gives a circuit level schematic for the simplest adiabatic logic gate, the buffer-inverter, along with its schematic symbol. High fidelity models of the buffer-inverter



(a) Circuit (b) Schematic symbol

Figure 2: Buffer-inverter adiabatic logic circuit

unsurprisingly also require high fidelity models of transistors modeled by nonlinear differential equations. Describing the adiabatic circuits in terms of its logical modes of operation requires several simplifying assumptions. First, transistors operate as simple switches that are either open or closed depending on the gate voltage. The circuit schematic includes two types of transistors, NMOS (Q_3 & Q_4) and PMOS (Q_1 & Q_2) making the fabrication of adiabatic logic circuits compatible with traditional CMOS circuits. The transistor has three terminals: gate, source, and drain. For NMOS transistors, when a voltage across the gate and source, V_{GS}, exceeds a characteristic threshold voltage, the transistor turns on. PMOS transistors operate similarly with polarities reversed. To simplify interpreting the circuit models that follow, the NMOS transistors are on when $V_{\mathrm{GS}}^{\mathrm{NMOS}} > 0V$ and the PMOS transistors are on when $V_{\mathrm{GS}}^{\mathrm{PMOS}} < 0V$. Second, when transistors are on, they have a constant characteristic resistance R_{on}. Third, transistors have zero leakage currents when off. Fourth, all parasitic capacitances and resistances are ignored. Fifth and finally, only the transistor's gate capacitance is considered. The different simplified modes of transistor operation are shown in Figure 3. Applying the transistor models in Figure 3 to Figure 2(a) for A='1', the different modes of operation are illustrated in

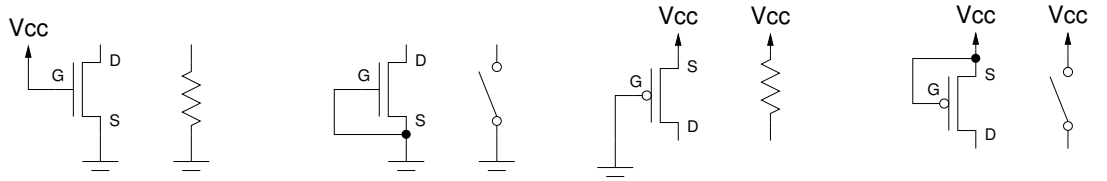


Figure 3: Simplified transistor operational modes

Figure 4. Note that C_F and $C_{\overline{F}}$ are the lumped capacitances for the circuit fanout. More concretely observed from the figure is the circuit charging and charge recovery in Figures 4(c) & 4(e) with no power consumed in Figures 4(b) & 4(d).

2.3 Adiabatic Combinational Circuit Architecture

The architecture of adiabatic combinational logic circuits is organized into several consecutive layers of logic powered by power clocks that are shifted in phase to facilitate the transfer of signal values from one layer to the next. Unlike traditional CMOS circuits where a wire conveys the logical signal, adiabatic circuits require two wires that operate in a complementary fashion when the circuit's power clock is operating in any segment except idle. This manner of signal is often referred to as dual-rail. Furthermore, the evaluation of adiabatic logic gates is synchronous with respect to its power clock unlike CMOS circuits which are entirely asynchronous. Indeed, adiabatic combinational logic circuits can be naturally pipelined with new inputs accepted at suitable times during the respective power clock phase. Figure 5 presents the architecture of a simple multilayered adiabatic logic circuit consisting of four buffer-inverters powered by four power clocks operated at phases displaced by 90°. Note transfer of values from one layer to the next when the current power clock is in the hold segment while the subsequent layer is in the charge segment.

3 Adiabatic VHDL Models

Refining the ideas introduced in §2, VHDL models for adiabatic circuits are presented here. Recall, hardware description languages (HDLs) are programming languages that model complex digital systems and are the source for hardware synthesis. HDLs facilitate specification of digital systems by modeling systems logically rather than at the circuit level. In addition, HDLs include familiar high-level language (HLL) programming capabilities for computing static constants, to support system modeling, and to provide desired modeling capabilities. For adiabatic logic circuits modeled at a temporal resolution where the influence of the power clock is important, the HLL features will be used to implement the logical behaviors of adiabatic logic circuits.

3.1 Anatomy of a VHDL Model

A VHDL model consists of an entity and an architecture [5]. The entity defines the model interface including the entity's signals, signal types, and signal modes (input, output, etc.). Further, model meta-information can be passed through optional generic parameters. Not unexpectedly, VHDL built-in types include the `bit` and `bit_vector` types. In addition, IEEE Standard 1164 [4] defines the more comprehensive `std_logic` type that better models use cases that occur in traditional digital circuits. For example, the `std_logic` type handles high impedance connections and wired logic connections for passive logic that are circuit level effects that extend to logic circuits. Considering the operation of adiabatic circuits described in §2, neither `bit` nor `std_logic` provides suitable models for adiabatic logic circuits.

Figure 6 shows an example of a VHDL model for a two-input AND gate using the `std_logic` type. The AND gate model shows declarations consisting of two inputs & one output and includes the behavioral model code for a two-input AND gate. Delays, extracted using a separate circuit analysis process, can be inserted consistent with the synthesized circuit.

3.2 Adiabatic Logic Values

At the circuit level, adiabatic logic values are more complicated than traditional logic values for several reasons. First, in adiabatic circuits, gate outputs are "dual railed" where a circuit structure generates both the true and complementary output values. Second, due to the effect of the power clock, the circuit output value is only valid at certain times as previously shown in Figure 5. Indeed, a logic one is a pulse that coincides with the circuit's power clock on the true sense gate output and a logic zero is a pulse on the complementary sense gate output while the circuit's power clock is active. While this operation is inherently analog, the circuit outputs can be categorized as logic one and logic zero. Taking a broader view of timing and circuit state, a suitable discretization of the behavior can be proposed in a manner that is consistent with adiabatic circuit operation. What follows is a discussion of the discretizing of the timing and circuit logic values.

The nature of the power clock provides straightforward guidance for discretizing time. With the dynamics of adiabatic circuits naturally falling into four distinct operating modes, it

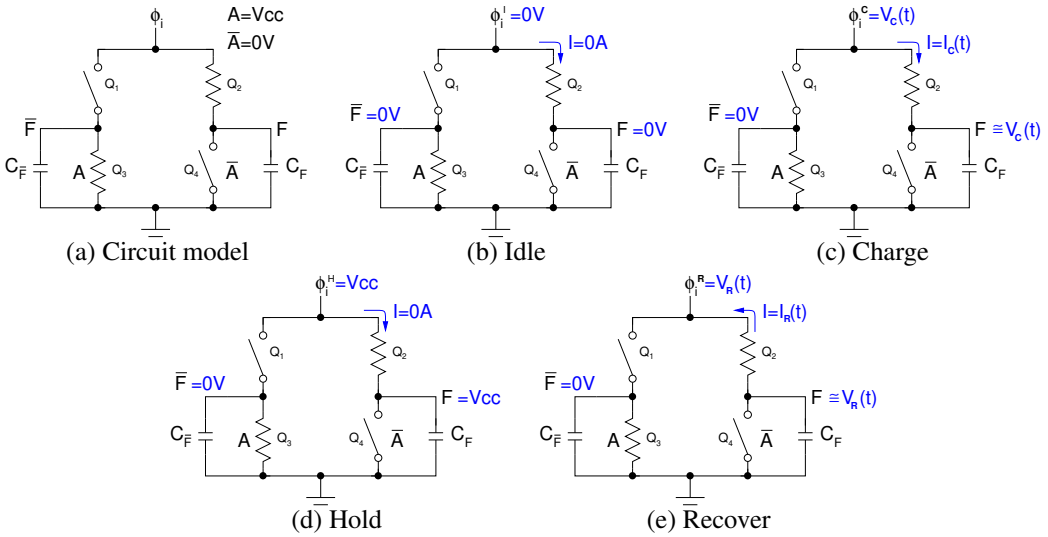


Figure 4: Simplified buffer-inverter circuit models for input A='1'

makes sense to discretize the phase into four segments. The following type declaration reflects the discretization suitable for adiabatic VHDL models.

```
type simplePhaseSegment is
            ('I','C','H','R');
```

The `simplePhaseSegment` type specifies the values `'I'`, `'C'`, `'H'`, and `'R'`, representing idle, charge, hold, and recover respectively. For segments where the power clock is changing (in `'C'` and `'R'`), no circuit dynamics are modeled, rather the logical result reflecting the values at the end of the segment are reported. Any varying circuit level quantities will be represented symbolically in that segment. Extending `simplePhaseSegment` is `phaseGeneral` which is a record including a `simplePhaseSegment` and phase index fields.

A new basic type, `aBitSimple`, is an eleven valued logic system defined to represent the range of adiabatic signal values that reflect the logic value, nature of the circuit, and value in relation to the phase. In this work, we have chosen to not differentiate the signal strengths during the charge and hold phases to facilitate interpretation of timing diagrams. Including these are straightforward and results in five additional signal values covering respective activities during the charge phase. The permissible values for this type are summarized in Table 1.

The fully qualified signal VHDL model is defined record type that includes both the signal value and the phase:

```
type aBit is record
  val     : aBitSimple;
  myPhase : phaseGeneral;
end record;
```

Including the phase in the signal definition enables run time checking to confirm the `aBit` phase is consistent with the assigned phase of the gate's power clock.

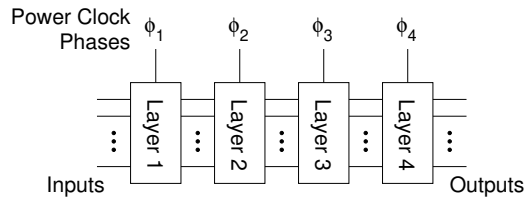
Several utility routines have been created to help manage signal values and phases. Some routines facilitate the conversion between standard signal types (`bit` and `std_logic`) and the new `aBit` type. Furthermore, operator overloading for the new logic type has been implemented to permit the natural composition of logic expressions. In the event indeterminate inputs or phase errors occur, the logic operations evaluate to unknown values (`'X'` or `'x'`) to facilitate troubleshooting. Finally, the logic values `'Z'` and `'z'`, along with the requisite bus resolution functions, permit high impedance bus modeling.

Table 1 Summary of adiabatic signal values for the `aBitSimple` type

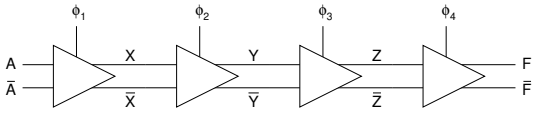
Value	Description
`'U'`	driving uninitialized value
`'X'`	driving unknown value
`'0'`	driving logic zero
`'1'`	driving logic one
`'Z'`	high impedance
`'u'`	recover uninitialized value
`'x'`	recover unknown value
`'L'`	recover logic zero
`'H'`	recover logic one
`'z'`	recover high impedance
`'*'`	fully discharged

3.3 Logical Adiabatic Gate Model

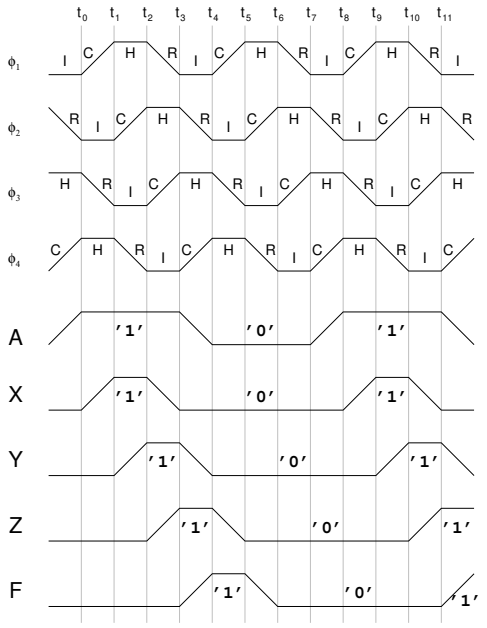
The logical adiabatic gate model requires changes both to the gate entity and to the behavior defined in its architecture compared with conventional gate models. The adiabatic gates perform logic functions, so one reasonable approach would be to adopt traditional logic values in the gate model. In this approach, phase information would be lost. Furthermore, adiabatic gates are dual rail, whose representation is not as important as the power clock phase in logical modeling.



(a) Circuit architecture



(b) Buffer-inverter chain



(c) Timing diagram for buffer-inverter chain.

Figure 5: Adiabatic circuit architecture and operation

```
entity and2 is
  port(a,b: in std_logic;z:out std_logic);
end entity and2;
architecture behavioral of and2 is
begin
   if a='0' or b='0' then       z<='1' after 500 ps;
   elsif a='1' and  b='1' then  z<='0' after 200 ps;
   else                         z<='X' after 350 ps;
   end if;
end architecture behavioral;
```

Figure 6: Example VHDL model

However, their explicit inclusion provides an opportunity to have visibility of all signals in the circuit. Apropos, the entity for the AND gate shown in Figure 7 includes dual rail input & output logic signals and the clock phase driving the gate.

```
entity adbAnd2 is
  port(
       phi : in generalPhase;
       a,an: in aBit;
       b,bn: in aBit;
       z,zn:out aBit
  );
end entity adbAnd2;
```

Figure 7: Entity for two-input adiabatic AND gate

Determining the gate outputs is no longer a simple matter of evaluating the gate's logic function based on the circuit inputs because of the dependence on the power clock segment. The model presented in Figure 8 implements the behavior for the two-input adiabatic AND gate that accounts for the power clock. When the clock phase changes, the gate inputs are verified to be in phase and to be correctly lagging the gate's power clock phase. When a phase error is detected, the output signal is assigned an `'X'` value. Since logical operations have been overloaded, the gate logic function is expressed in a natural fashion, permitting logic equations to model the respective MOS switching networks. Logic operations are evaluated in their respective common phase, facilitating the composition of complex logic functions. The resulting output value is stored in a temporary variable so that the phase can be correctly updated to be consistent with the power clock for the gate. In transitioning to and during the hold segment, the logic gate outputs remain constant in the model.

3.4 Extending to Other Logic Gates

The dual rail nature of the logic gates simplifies creating families of logic gates. Signal inversion is accomplished simply by swapping the true and complementary signal rails requiring no additional circuitry. Indeed, with DeMorgan's Theorem, it is easy to show that by swapping dual rail signals to complement inputs & outputs, the two-input AND gate can also serve as an

Table 2 Utility functions and procedures

Name	Purpose
isCharging	function, returns true when power clock is charging
isHolding	function, returns true when power clock is maximal
isRecovering	function, returns true when power clock is discharging
isIdle	function, returns true when power clock is off
deenergize	procedure, reduces the strength of signal while retaining logic value
assignToPhase	procedure, assigns a phase to a signal

```
process(phi)
  variable zInt,znInt:aBit;
begin
  -- check for valid input and output
  -- phase segments
  if(isCharging(phi)) then
    zInt  <= a  AND b;
    zIntn <= an OR  bn;
  elsif isHolding(phi) then
    -- by VHDL semantics,
    -- no update-no signal change
  elsif isRecovering(phi) then
    zInt  := deenergize(zInt);
    znInt := deenergize(znInt);
  else -- idle
    zInt.val  := '*';
    znInt.val := '*';
  end if;
  assignToPhase(zInt, phi);
  assignToPhase(znInt, phi);
  z  <= zInt;
  zn <= znInt;
end process;
```

Figure 8: Behavioral model for two-input adiabatic AND gate

OR, NAND, or NOR gate. In addition, more complex logic functions can be modeled using the logic equation for the true input values and the dual logic equation for the complementary input values.

For example, the logic equations for a full adder are

$$\begin{aligned} S &= A \oplus B \oplus C_i \\ C_o &= A \cdot B + A \cdot C_i + B \cdot C_i \end{aligned} \tag{1}$$

With traditional CMOS logic, the full adder can be implemented with several gates. In adiabatic logic, each logic function can be implemented with an NMOS switching network, so the full adder can be implemented with two adiabatic logic gates. The logic equations for the complementary networks are

$$\begin{aligned} \overline{S} &= \overline{A} \oplus \overline{B} \oplus \overline{C}_i \\ \overline{C}_o &= (\overline{A} + \overline{B}) \cdot (\overline{A} + \overline{C}_i) \cdot (\overline{B} + \overline{C}_i) \end{aligned} \tag{2}$$

The second example is a multiplexer with dedicated, mutually exclusive select lines. The general true and complementary logic equations are

$$Z = \sum_{i=0}^{N-1} S_i D_i \qquad \overline{Z} = \prod_{i=0}^{N-1} (\overline{S}_i + \overline{D}_i), \tag{3}$$

where N is the number of data inputs. It is also easy to show that, for $N = 2$, (3) can specify two-input XOR and XNOR gates.

3.5 Test Bench

A test bench is a special VHDL model which is used to verify the circuit model. The test bench instantiates the unit under test, generates all stimulus, and can include code to verify the model's outputs are correct. Figure 9 gives the VHDL process that generates the i^{th} power clock. For four power clock phases, each power clock phase i has the same period T and is delayed by $(i-1) \times 90^\circ$, or $T(i-1)/4$ with respect to a reference time at the start of the simulation. This can be easily generalized for a different number of power clock phases. In Figure 9, the power clock process includes one full clock period interval at the beginning of the simulation with no activity among all clocks. The first `wait` statement ensures that all power clocks are inactive for at least one full period of the power clock and the start of each is delayed to ensure each clock will be in the appropriate relative phase.

```
...
constant T: time := 100 ns;
...
process
-- generate the ith power clock phase
-- i in {1,2,3,4}
begin
  Phi_i <= ('I',i-1);
  wait for T*(3+i)/4; -- See narrative
  loop
    Phi_i.segment<='C'; wait for T/4;
    Phi_i.segment<='H'; wait for T/4;
    Phi_i.segment<='R'; wait for T/4;
    Phi_i.segment<='I'; wait for T/4;
  end loop;
end process;
```

Figure 9: Generating the i^{th} phase of the power clock

In order for outputs to conform to proper adiabatic operation, inputs must be set in the appropriate manner to ensure the adiabatic operation of the gate receiving the input. In addition, it is possible that different inputs may be required at different logic layers, and hence must be synchronized to the correct power clock phase. This can be accommodated in one of two ways. First, the inputs can be provided at the same time and always on the same phase. In this case, buffers will need to be inserted to delay the signal until it has the required phase for its respective input layer. Second, the inputs can be provided and synchronized to the required phase. The modeling satisfies either case.

4 Examples

Three examples of adiabatic logic circuit models are presented here. In the first, a full adder model is presented. In the second, a Kogge-Stone adder model is presented. In the third, a more complex model of the AES S-Box is presented. The models were verified using GHDL Version 0.33 under the IEEE-1164 1993 release of the VHDL standard on Ubuntu 16.04. In addition, while the modeling is based on the 1993 standard, no issues are anticipated for later VHDL standard releases. Waveforms are displayed using the GTKWave V3.3 waveform viewer.

4.1 Full Adder

A simple but useful example to consider is the full adder. The full adder is a key building block used to implement computer arithmetic hardware. The full adder model consists of two logic gates and operates using one power clock phase using the logic functions defined in (1) and (2). The behavior is modeled by modifying the code in Figure 8 by substituting the logic equations for the sum and carry functions respectively in place of the AND gate logic equations. The simulation results are presented in Figure 10. The inputs provided to the full adder sequence through all eight input combinations in successive power clock cycles, noted with cursors A-H.

4.2 Kogge-Stone Adder

The next example is a Kogge-Stone adder (KSA) [2, 6] and demonstrates the operation of a more complex multilayered combinational circuit. The KSA adds two binary integers and is among the fastest combinational adders, whose implementation requires log_2N+2 layers of adiabatic logic. The KSA adder can be fully implemented with commonly known gates such as AND, OR, XOR, & etc. By implementing certain composite functions to provide carry generates & propagates as individual logic gates, the circuit architecture can be simplified. Indeed, these composite gates are part of the formulation of KSA adders and are summarized in Table 3. Note that the Buffer cell is not a part of the traditional KSA adder formulation. Rather, the Buffer cell is included in this model to support proper adiabatic circuit operation to match the power clock phase for the values propagating from layer to layer in the adder. Note also that subscripts on gate input values are nominal and are related to the local interconnections required to implement the KSA adder.

Table 3 Kogge-Stone logic cells

Cell	Logic Equation	
Black cell	$G_{\text{out}} = (P_1 \cdot G_0) + G_1$	$P_{\text{out}} = P_1 \cdot P_0$
Gray cell	$G_{\text{out}} = (P_0 \cdot G_0) + G_1$	
White cell	$G_{\text{out}} = P_1 \cdot P_0$	$P_{\text{out}} = P_1 \oplus P_0$
Buffer cell	$G_{\text{out}} = G_0$	$P_{\text{out}} = P_0$

The VHDL model for the KSA adder has been implemented in a generic fashion so that the same architecture can implement any power-of-2 sized KSA adder. Figure 11 gives the entity used to model the KSA adder. In order to simplify the presentation of results, a four-bit KSA adder is demonstrated. Specifically, the VHDL model for the four-bit adiabatic KSA adder modeled here requires $log_2N+2=4$ layers of logic to implement. Figure 12 shows the simulation beginning at 4 μs for a circuit powered by power clocks with 100 ns periods. Note that signal complements have been omitted. At cursor A (4.0875 μs), the input $Op1_A$=0101, $Op2_A$=1001 and Cin_A=1. The output layer is charging at cursor C (4.1875 μs) and $Cout_C$=0, and Sum_C=1111. In addition, at cursor C, the inputs are changed to $Op1_C$=1010, $Op2_C$=0101 and Cin_C=0 resulting in $Cout_F$=0, and Sum_F=1111 at 4.2875 μs.

4.3 Advanced Encryption Standard (AES) Substitution Box

In this section, we present a significantly more complex model which is the logic for the Advanced Encryption Standard (AES) substitution box (S-box). The purpose of the S-box is to introduce a nonlinear, but difficult to reverse, transform to enhance the security of the encryption. The interested reader can find more information by consulting the AES standard [12]. The S-box is a complex combinational logic function devised by others [11, 13]. Their proposed circuit, however, cannot be directly implemented using adiabatic logic circuits because of the multiphase nature of the adiabatic logic circuits.

The logic for the S-Box follows from the implementation method proposed by Satoh et al. [13] and detailed combinational logic S-Box implementation described by Mui [11]. The respective authors note the efficiency of their implementation in terms of hardware. Figure 13 gives an overview of the S-Box and inverse S-Box logic. A transformation, (δ), is applied to the original $GF(2^8)$ system to permit decomposition in terms of a $GF(2^4)$ system, and subsequently a $GF(2^2)$ system to permit derivation of logic functions for intermediate values [11, 13]. Once the system is expressed in terms of a $GF(2^2)$ system, the logic functions at this level can be expressed directly as four-input, two-output logic expressions. From the $GF(2^2)$ logic functions, logic expressions for the $GF(2^4)$ and then ultimately $GF(2^8)$ can be derived.

Figure 14 shows the individual logic blocks that are used in both the S-Box and inverse S-Box transformations. The number of layers of adiabatic logic are indicated above each block. The logic is mostly implemented with two-input gates along with a handful of three-input gates. The δ and affine transforms T are matrix/vector operations on individual bits using AND and exclusive-OR operations, i.e. $GF(2)$.

Inspection of Figures 13 & 14 reveal a complex hardware organization with a multitude of paths for logical results that flow through varying numbers of layers of logic. For proper operation, the phases of inputs received for any block must be identical and the block must be energized by the next sequential

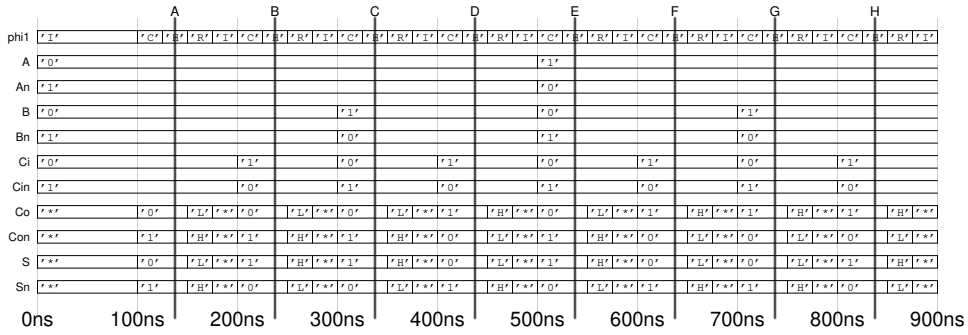


Figure 10: Full adder simulation results

```
entity KsaGeneric is
  generic(order : integer := 2); -- width=2**2=4
  port (
    phi1,phi2,phi3,phi4 : in  phaseGeneral;
    adbCin,  adbCinN    : in  aBit;
    adbOp1,  adbOp1n    : in  aBit_vector(2**order-1 downto 0);
    adbOp2,  adbOp2n    : in  aBit_vector(2**order-1 downto 0);
    adbSum,  adbSumN    : out aBit_vector(2**order-1 downto 0);
    adbCout, adbCoutN   : out aBit
  );
end KsaGeneric;
```

Figure 11: Entity for Kogge-Stone adder. Note that ** has been overloaded for integer types

power clock phase. Figure 15 gives the annotated block diagram for the adiabatic logic implementation. Differing from previous examples, the logic circuit is implemented using six power clock phases so that complementary power clock phases, power clocks exactly 180° out of phase, are nonoverlapping. Further, because the input phases to a logic block must match, the phase for an unmatched signal is matched with its destination by adding a suitable number of buffer-inverter gates and are denoted by the Φ^N blocks. The solution attempts to optimize the hardware by not fully pipelining the forwarding in some cases.

Both the S-Box and inverse S-Box models were simulated for all possible input combinations and verified for correctness in the test bench. Figure 16 gives an example timing result for an S-Box input of `11000011`.

5 Summary and Future Work

A modeling framework has been presented that is consistent with the logical operation of adiabatic logic circuits. A new type, `aBit`, was defined that captures the main modes of operation for adiabatic circuits. The type models the principle adiabatic signal features and ties the operation of the logic circuits to the power clock. The framework for defining logic functions was presented. Finally, three modeling examples with their respective simulation results were presented.

Future work will include verifying the operation of the modeling framework on a wider variety of adiabatic and reversible circuits. In addition, applicability to different clocking schemes & timing, energy modeling, and transistor level synthesis will be investigated as well.

References

[1] E. Christen and K. Bakalar. "VHDL-AMS – A Hardware Description Language for Analog and Mixed-Signal Applications." *IEEE Transactions on Circuits and Systems–II Analog and Digital Signal Processing*, 46(10): 1263–1272, October 1999.

[2] M. Cutitaru. *IDPAL A Partially-Adiabatic Energy-Efficient Logic Family: Theory and Applications to Secure Computing*. PhD Thesis, Old Dominion University, Norfolk, Virginia, USA, August 2014.

[3] J. S. Denker. "A Review of Adiabatic Computing." *IEEE Symposium on Low Power Electronics*, 94–97, San Diego, California, USA, pp. 94–97, September 1994.

[4] IEEE Computer Society. *IEEE Standard Multivalue Logic System for VHDL Model Interoperability (Std_logic_1164)*, March 1993.

[5] IEEE Computer Society. *IEEE Standard VHDL Language Reference Manual*, IEEE Std 1076™-2008, January 2009.

[6] P. M. Kogge and H. S. Stone. "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations." *IEEE Transactions on Computers*, C-22(8): 783–791, August 1973.

[7] J. Koller and W. Athas. "Adiabatic Switching, Low Energy Computing, and the Physics of Storing and Erasing Information." *Workshop on Physics and Computation, 1992. PhysComp '92*, Dallas, Texas, USA, pp. 267–270, October 1992.

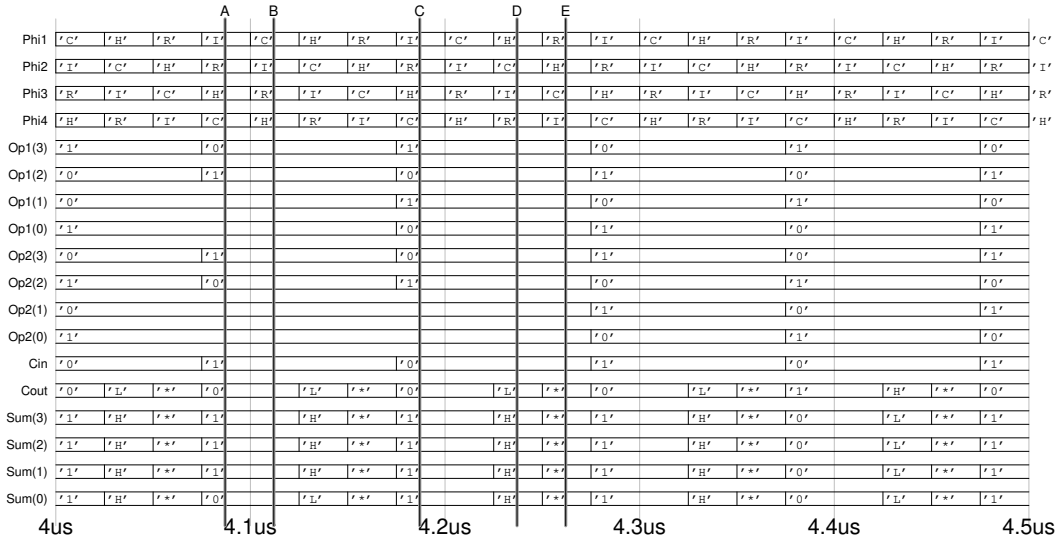
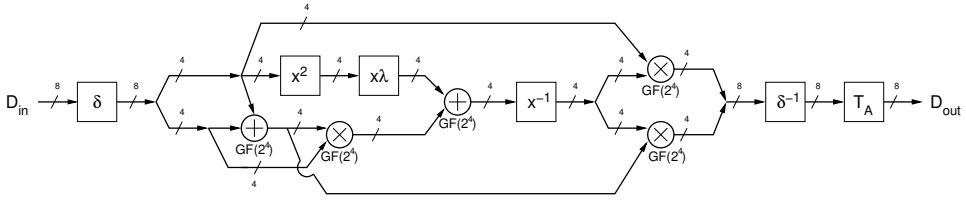
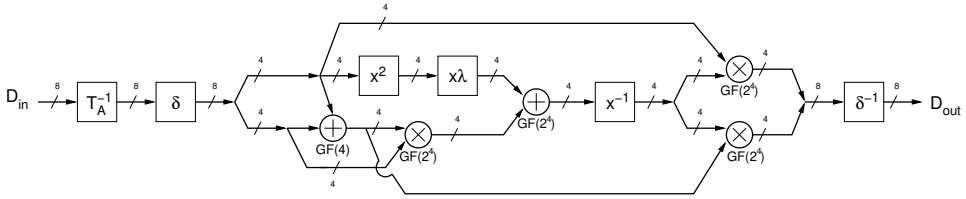


Figure 12: KSA Simulation results at 4us



(a) Forward S-Box transformation



(b) Inverse S-Box transformation

Figure 13: S-Box block diagram based on composite field decomposition [11, 13]

[8] A. Kramer, J. S. Denker, B. Flower, and J. Moroney. "2nd Order Adiabatic Computation with 2n-2p and 2n-2n2p Logic Circuits." *Proceedings of the International Symposium on Low Power Design ISLPD'95*, Dana Point, California, USA, pp. 191–196, 1995.

[9] R. Mita and G. Palumbo. "Modeling of Analog Blocks by Using Standard Hardware Description language." *Analog Integrated Circuits and Signal Processing*, 48(2):107–120, August 2006.

[10] Y. Moon and D.-K. Jeong. "An Efficient Charge Recovery Logic Circuit." *IEEE Journal of Solid-State Circuits*, 31 (4):514–522, April 1996.

[11] E. N. Mui. "Practical Implementation of Rijndael S-Box Using Combinational Logic." unpublished technical report, 2007.

[12] NIST. "FIPS PUB 197, Advanced Encryption Standard (AES)," U.S. Department of Commerce/National Institute of Standards and Technology, 2001.

[13] A. Satoh, S. Morioka, K. Takano, and S. Munetoh. "A Compact Rijndael Hardware Architecture with S-Box Optimization." *7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2001)*, Gold Coast, Australia, pp. 239–254, December 2001.

[14] L. Varga, G. Hosszú, and F. Kovács. "Two-level Pipeline Scheduling of Adiabatic Logic." *International Spring Seminar on Electronics Technology (ISSE 2006)*, St. Marienthal, Germany, pp. 390–394, May 2006.

[15] D. J. Willingham. *Asynchrobatic Logic for Low-Power VLSI Design*. PhD thesis, University of Westminster, London, England, March 2010.

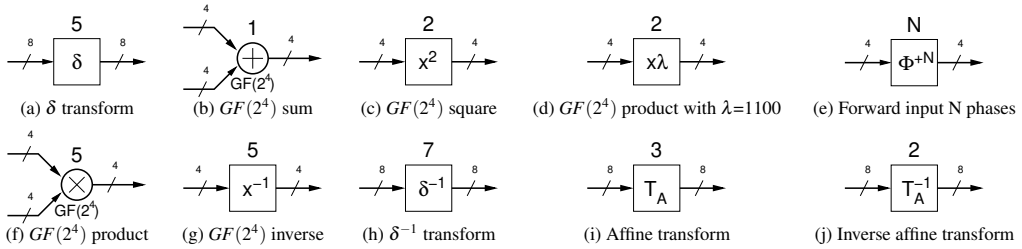


Figure 14: S-Box block components. The number above each component is its respective number of logic layers.

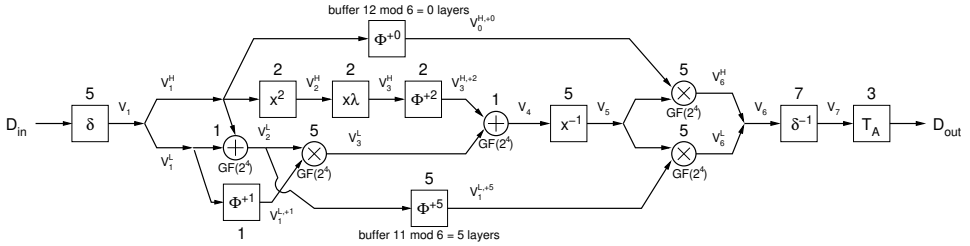


Figure 15: Annotated S-Box block diagram.

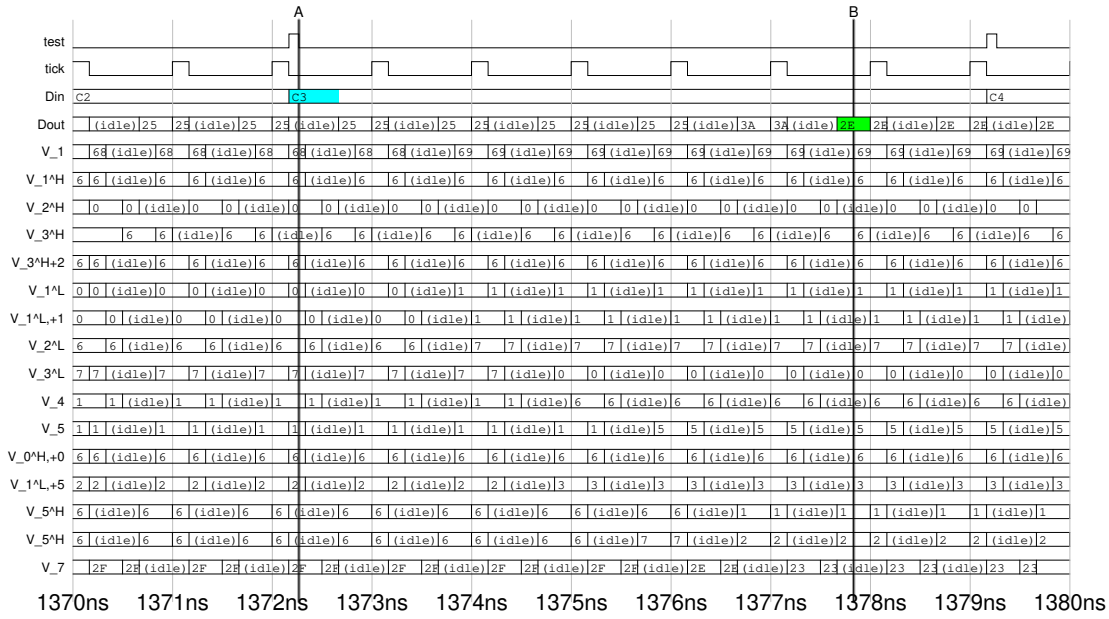
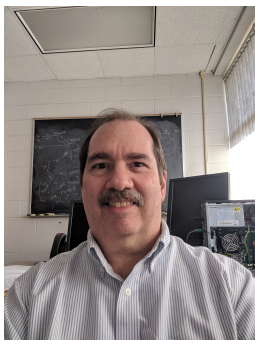


Figure 16: S-Box simulation result for an input of `11000011`



Lee A. Belfore II joined the Department of Electrical and Computer Engineering at Old Dominion University, Norfolk, Virginia, in 1997 and is currently an Associate Professor. He received his Ph.D. in Electrical Engineering from The University of Virginia, his MSE in Electrical Engineering/Computer Science from Princeton University, and his BSEE in Electrical Engineering from Virginia Tech. His research interests include modeling and analysis of low power digital electronics technologies, custom data processing systems using FPGAs, machine learning, and autonomous vehicles.

Mining for Causal Regularities

Thomas Bidinger*, Hannah Buzard*, James Hearne*, Amber Meinke*, and Steven Tanner*
Western Washington University, Bellingham, Washington 98225, USA

Abstract

This paper reports on an algorithmic exploration of the theory of causal regularity based on Mackie's theory of causes as MINUS conditions, i.e., minimal insufficient but necessary member of a set of conditions that, though unnecessary, are sufficient for the effect. We describe the algorithm to extract causal hypotheses according to this model and the results of its application to a number of real-world data sets. Results suggest further promising applications, modifications and extensions that might derive further insights of a dataset.

Key Words: Causal regularity, data mining, INUS condition, MINUS condition, Mill's methods.

1 Introduction

Of the several established approaches to the notion of causality, the regularity view is the oldest. It was introduced by Hume in the 18th century, elaborated upon by Mill in the 19th century and finds its most detailed articulation in Mackie in the 20th century. In this view, causes are to be identified as conditions or events that are uniformly accompanied or followed by some effect. Importantly, in this view, no other intrinsic relation between cause and effect is assumed other than regularity. This theory of causes invites the possibility of a search for causes by appeal to strict pattern matching, independent of statistical or probabilistic considerations. What is reported here is one approach to realizing this conception of causes and their identification.

2 Approach

In the more recent formulation by Mackie, the causal antecedents of an effect are complex configurations of facts. To motivate this view, note that a match might flare because it is struck on an abrasive surface in the right conditions – absence of moisture and presence of oxygen – or it might flare because it is heated to a flash point under similar conditions, or placed in proximity to another flame. In Mackie's formulation, a cause is what he dubs an INUS condition, an insufficient but necessary member of a set of conditions which, though unnecessary, are sufficient for the effect. Formally, this means that the search for causes is equivalent to searching for valid implications whose right-hand side is the effect and whose left-hand side is a disjunctive normal form expressing configurations of conditions: There is no a priori restriction on the number of elements in each of the conjuncts nor any restriction as to their number. Indeed, there are several other variations on this idea, as well as extra constraints discussed below. A further constraint, not originally articulated by Mackie, is that the conjunctions participating in an INUS be minimal; that is, they should be purged of unnecessary conjuncts.

3 Related Work

There is a large literature on causal discovery. Since this research concerns the causal regularity theory of causation, we restrict or review antecedents in the literature to work in that tradition. Although initiated by Hume and elaborated upon by J.S. Mill, the current *locus classicus* of the regularity theory is the article by (Mackie, 1964) [5], followed by his monograph (Mackie, 1984) [6]. Since then, Mackie's view has received a number of computational treatments. Baumgartner has provided philosophical justification of the regularity theory of causation and developed an algorithm restricted to configurations of Boolean values (Baumgartner, 2009 [2], 2009). Beirlan, Leuridan and Van De Putte support the idea computationally through a decidable subset of first order logic (Bierlan 2018) [3].

4 Method

4.1 A Brief Description

The purpose of our algorithm is to find cause-effect relationships implicit in datasets. The datasets used must consist of a table where each row is one observation, and each column represents an event. Each cell can signify that an event occurred, didn't occur (negated event) or that it's unknown whether or not it occurred. The algorithm starts with a chosen event and creates conjunctions (lists) of all events that also occur in the case that the chosen effect also occurred. Each conjunction that was generated is then tested to see whether it is necessary to the chosen effect occurring or not. A conjunction is necessary if it is not a superset of any of the other conjunctions. If it is a superset, then it is removed because there

* Department of Computer Science, 516 High Street. Emails: bidingt@wwu.edu, buzardh@wwu.edu, James.Hearne @wwu.edu, meinkea@wwu.edu, and tanners2@wwu.edu.

exists a simpler conjunction that better represents the data. These new conjunctions are then tested for sufficiency. Sufficiency is tested by ensuring that the chosen effect MUST occur if the given set of conjuncts also occurs. This is done by making sure that the set of conjuncts does not also appear in any of the rows where the chosen effect does not occur. Adding Figures and Tables.

4.2 In-Depth Account

4.2.1 Input Dataset: The algorithm assumes an ontology of individual objects and a collection of predicates which may or not be true of each. The algorithm also accommodates worlds in which predicate values are unknown for some objects. Assuming that the predicates will be relatively few and in order to accommodate an indefinite number of objects, it happens that columns correspond to predicates/attributes and rows denote the value of the predicates when applied to each object, including the possibility of unknown true values.

4.2.2 Formatting the Input Dataset: The dataset that is given as input is formatted according to how the algorithm expects the data to appear (only values of 1, 0, and -1 are accepted). This allows our algorithm to be versatile and make accurate computations for all datasets. The program prompts the user to label each column and choose whether to keep the column data as it is, remove the column from the dataset, or one-hot encode the data in the column. One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms using 1's and 0's. A visual representation of how to utilize one-hot encoding is shown in Diagram 1 below. The user is prompted to make these changes because our algorithm will not accept data that is not able to be represented with a 1, -1, or 0 and will ask the user to re-input the data if it finds a number not in this form.

4.2.3 Choose an Effect: The effect whose possible causes is in input to the process. The data is separated into two subsets: one object for which the effect is positive and one for objects in which the effect is negative, i.e., conditions in which it does not occur. The algorithm uses the former subset to generate potential MINUS-conditions, and the latter subset to check whether or not the generated potential MINUS-conditions are sufficient to prove the chosen effect.

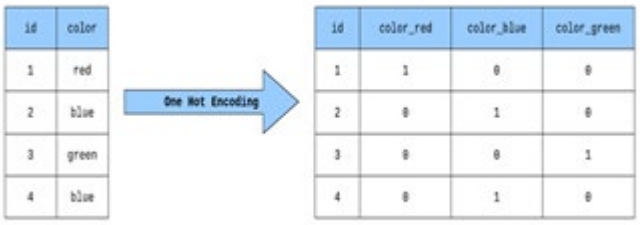


Diagram 1

4.2.4 Generate Possible Minus Conditions. To generate potential MINUS-conditions, the algorithm iterates through all rows in the data where the chosen effect obtains. For each of these rows, a set is created of all of the predicates in the row. All subsets are generated using this row data and each set that doesn't include the chosen effect and is not empty is added to the set of potential MINUS-conditions. It should be noted that this set of generated MINUS-conditions is a superset of the set that contains all sufficient conditions, and that the disjunction of the potential MINUS-conditions in this set is necessary for the chosen effect to occur (assuming there is more than one event type).

4.2.5 "Sufficient" and "Necessary" MINUS-condition check: Once the algorithm has generated the set of possible MINUS-conditions, each condition needs to be tested to check that it is sufficient and not a superset of a previously identified condition. By removing such supersets, the algorithm ensures that all parts of the proven MINUS-conditions are necessary. In order to prove that a given potential MINUS-condition is sufficient, the algorithm checks if the entire conjunction occurs in a row where the effect does not occur. Given that the conjunction being tested was previously known to occur when the chosen effect occurs, then if it also does not occur in a row where the chosen effect is negative, then it is proven to be sufficient for the given dataset.

4.2.6 Algorithm Output: Finally, all the identified MINUS-conditions are combined into a disjunctive normal form, presented as output. Each conjunction (MINUS-condition) is considered as a causal for the effect., one element of which would be what is normally identified as the cause of the event. It is important to note that there may be a case where it is impossible to have a necessary disjunction of conjunctions for a specific chosen effect (see Table 1 using R as the chosen effect). In Table 2, all data for rows 'a' and 'c' are equivalent except for the effect which occurs in one and doesn't occur in the other. Because of this, there is no MINUS-condition that includes row c. Therefore, no disjunction of MINUS-conditions exists which is necessary for R to occur in this dataset. However, if there is a disjunction of MINUS-conditions where the disjunction is necessary, each disjunct is sufficient, and each conjunct in each conjunction is necessary for the conjunction to be sufficient, then the algorithm will give this as output.

Table 1

	P	Q	R
a	1	-1	-1
b	1	1	1
c	1	-1	1

Table 2

	P	Q	R	S	T	U
a	1	-1	-1	0	1	-1
b	1	0	1	1	-1	1
c	-1	-1	0	-1	1	-1
d	1	-1	-1	-1	0	1
e	1	0	1	-1	-1	1
f	1	-1	-1	0	1	0
g	0	1	-1	-1	-1	0
h	1	-1	1	1	-1	1

5 Datasets

The datasets used for our algorithm contain columns which refer to events/attributes and rows that refer to observations. The value for an observation can be either 1 (event occurred), -1 (event did not occur), or 0 (unknown). Our algorithm is unique from other algorithms in the sense that it allows for the usage of a 0 or unknown in the dataset.

The following is an example of a dataset appropriate to our implementation that has been used in the development of this algorithm. In this case, considering the event 'P' we can see that all observations occurred except for the observation 'c' and there is no data provided for observation 'g.

A portion of the Cleveland Heart Disease dataset explored in greater depth below is shown in Table 3. This dataset is important to visualize and understand due to the fact that it can be used to gain actual knowledge and insight on the causation of a specific issue, unlike the dataset containing just letters above.

Table 3

Disease	Age	Sex	ind_typ_angina
1	-1	63	1
2	1	67	1
3	1	67	1
4	-1	37	1
5	-1	41	0
6	-1	56	1

The columns of the dataset (this is only 4 of the 14 we used) indicate whether the patient was afflicted with heart disease, their age, their gender, and whether they experience typical angina (chest pain) or not. To understand one column of the data, for the sex columns, all 5 of the patients in this subset were male and the gender is unknown for one patient, represented as a '0'. We used this dataset to find conjunctions of conditions that are shown to cause heart disease. These results are discussed in the 'Results and Analysis' section below.

6 Datasets

6.1 Urinary Disease Dataset Description

6.1.1 Dataset Description: This data was designed to automate the decision making/diagnosis of the presumptive diagnosis of two diseases of the urinary system, "Acute Inflammation of Urinary Bladder" and "Acute Nephritis of Renal Pelvis". The dataset contains six attributes applicable to these two diseases, which have similar, but not identical symptoms. These attributes are as follows: fever present (at or above 38C), occurrence of nausea, lumbar pain, urine pushing (continuous need for urination), micturition pains (pain while urinating), and urethra discomfort (burning, itching, or swelling). The dataset also contains information on whether or not each patient associated with this data has one of the diseases, no diseases, or both diseases. Each instance (row) in the dataset represents a patient.

6.1.2 Dataset Description: There are certain symptoms known by experts to signify one or the other disease. One test of the utility of our algorithm for identifying causal regularities is to see whether it replicates expert knowledge. Dr. Czerniak, of the Polish Academy of Sciences, reports that Acute Inflammation of the urinary bladder is characterized by sudden occurrence of pains in the abdomen region, urination in the form of constant urine pushing, micturition pains, and sometimes lack of urine keeping. The excreted urine is turbid and sometimes bloody. The body experiences a temperature rise, however most often not above 38C. By contrast, Acute nephritis of the renal pelvis begins with a sudden fever that reaches and sometimes exceeds 40C. The fever is accompanied by shivers and one-or both-side lumbar pains. Not infrequently, there is nausea and vomiting and spread of pains in the whole abdomen. Again, symptoms of acute inflammation of the urinary bladder appear very often. Our dataset does not cover all of these characteristics/symptoms such as gender, shivers, and entire abdomen pain, however, given the data we do have, we should be able to match up applicable attributes. The algorithm will give us the combination of individual conditions that lead to each respectable disease and should pair well with the human expert findings.

6.1.3 What the Algorithm Identifies: When the algorithm is run using "Inflammation of urinary bladder" as the chosen effect, we receive the following proven conditions:

('~fever', 'urethra-discomfort')
('~fever', '~lumbar-pain')
('~fever', 'urine-pushing')
('~lumbar-pain', 'urethra-discomfort')
('~nausea', 'micturition-pains')
('~lumbar-pain', 'urine-pushing')
('nausea', 'urethra-discomfort')
('~fever', 'micturition-pains')
('micturition-pains', 'urethra-discomfort')
('nausea', 'urine-pushing')
('~lumbar-pain', 'micturition-pains')
('urine-pushing', '~urethra-discomfort')
('urine-pushing', 'micturition-pains')

When the algorithm is run using "Nephritis of renal pelvis origin" as the chosen effect, we receive the following proven conditions:

('nausea',)
('~urine-pushing', 'micturition-pains')
('~micturition-pains', 'urethra-discomfort')
('fever', 'urine-pushing')
('fever', 'lumbar-pain')
('lumbar-pain', 'urine-pushing')

('fever', 'urethra-discomfort')
('fever', 'micturition-pains')
('lumbar-pain', 'urethra-discomfort')
('lumbar-pain', 'micturition-pains')

These results are very promising and replicate closely the expertly deduced symptoms. As the output shows, some symptoms such as urethral discomfort are present more or less in both diseases, however when these symptoms happen in conjunction with lumbar pain or a fever, this always signifies nephritis, not inflammation. Likewise, if a patient has these symptoms and no lumbar pain or fever, they almost certainly have inflammation of the urinary bladder. There are also some conditions, mainly micturition pains, which seem to be only slightly more characteristic of inflammation over nephritis.

6.2 Heat Disease Dataset

6.2.1 Dataset Description: A promising dataset that both exhibits the accuracy of our algorithm and reveals important information regarding heart health, is the Cleveland Heart Disease dataset from the UCI repository. The Cleveland dataset is one of the most used datasets in Machine Learning and can be used to classify whether an individual is at risk for suffering from heart disease or not. The data was retrieved from 303 individuals and originally contained 76 columns, however, the shorter version of the dataset, which has been used for all the published experiments, contains only 14 columns. These columns are the 14 attributes that were found to have the biggest impact on classifying heart disease. These attributes chosen for the Machine Learning experiments are: age, sex (male or female), whether the patient was experiencing typical angina, atypical angina, or non-angina related chest pain (angina is chest pain caused by reduced blood flow to the heart), resting blood pressure, serum cholesterol level, fasting blood sugar (should be less than 120 mg/dl), heart disease flagged on ECG 1, heart disease flagged on ECG 2, patient's max heart rate, whether the patient experienced exercise induced angina, patient's peak exercise ST segment upward slope indicator, patient's down slope indicator of peak exercise ST segment, number of patient's major vessels colored by fluoroscopy, whether the patient has reversible thalassemia defect or fixed thalassemia defect (thalassemia is an inherited blood disorder).

To get results that show the proven conjunctions of attributes that cause heart disease, we ran the algorithm and gave 'disease' as input for the chosen effect. The algorithm checked 93,759 conditions in 2 minutes and gave 96 proven conditions as output.

6.2.2 What We Expect: Choosing 'disease' as the chosen effect, we would expect to see conjunctions of the following as causes of heart disease based on research by medical professionals:

Sex: Males are more likely to develop heart disease than women

Blood Sugar: A fasting blood sugar is expected to be between 80-100, and anything over 100 can be an indicator of multiple diseases or illnesses, one of those being heart disease

ECG Indicators: If an ECG indicates abnormal heart rates or an abnormal pattern once or especially twice, these are most likely signs of heart issues such as heart disease

Thai Fixed and Reversed Defect: This is a type of blood issue that can lead to organ failure as well as heart issues (one of the issues being heart disease). Fixed means that it is permanent and reversed means that the defect can be reversed, however, both of them can still cause organ damage and heart failure.

Fasting blood sugar: Over time, high blood sugar can damage blood vessels and nerves that control the heart which can cause heart attacks

Max heart rate: Numerous studies have shown that higher resting heart rate is associated with increased risk of cardiovascular events and death in men and women.

Upward slope indicator: ST-Elevation is very serious and can mean that one of the heart's major arteries is blocked. Even if the artery is not currently blocked, any abnormal ST-Elevation indicates risk of major artery blockage

Down slope indicator: ST-segment depression is associated with a 100% increase in the occurrence of three-vessel/left main diseases and to an increased risk of subsequent cardiac events

Number of patient's major vessels colored by fluoroscopy: Fluoroscopy is used to help the healthcare provider see the flow of blood through the coronary arteries to check for arterial blockages. The more blood vessels that show blockage, the higher the patient is at risk for heart disease and heart attacks

Serum Cholesterol: A high serum total cholesterol level has been proven to indicate a potential increased risk for heart disease.

Resting blood pressure, exercise induced angina and typical/atypical angina alone cannot accurately predict heart disease or risk of heart attack, which is why they are not listed above. However, either of these in conjunction with each other or other symptoms can be a proven indicator of a patient being at risk.

6.2.3 What the Algorithm Reveals: In order to best understand the results we received, we have analyzed a subset of the output to ensure that the algorithm output matches what we would expect to see based on expert research. Every conjunction below lines up with what expert researchers would say could be a cause of heart disease. Each condition that is not listed below was also analyzed for correctness.

('sex', 'bloodsugar_exc120', 'ind_for_ecg_2', 'ind_exerc_angina')

This conjunction is valid for being a direct cause of heart disease according to the research above. Sex can be a cause of heart disease because males have been shown to be more prone to heart disease and heart attacks than women. Having an indicator for heart disease show up on an ECG is also typically shown to be accurate and mean a patient is at risk for heart

disease. Also, as I explained previously, an individual's blood sugar exceeding 120 and exercise induced angina cannot be a cause on its own, however in conjunction with each other and the other attributes, they can both be a proven cause.

('ind_atyp_angina', 'ind_exerc_angina', 'fixed_defect')

Using the information above, exercise induced angina and atypical angina cannot be a cause on its own but in this case both of those are paired together and also paired with the patient having thai fixed defect. Fixed thai defect is irreversible which means it usually eventually will lead to heart failure and in conjunction with also having chest pain occurring in multiple instances, this patient is rightfully flagged as being at risk for heart disease.

('ind_for_ecg_1', 'rev_defect')

Thai reversed defect on its own can be an indicator of heart disease even though it is reversible because it can cause organ damage to the heart. This defect paired with the patient being flagged for heart disease on their ECG puts the patients in an at risk category and the algorithm correctly identifies them as potentially suffering from heart disease.

6.3 Soybean Dataset

6.3.1 Dataset Description: One illuminating dataset is the one on soybean diseases from the 1980s by R.S. Michalski and R.L. Chilausky. The dataset concerns soybean disease diagnosis so we can use it to analyze the performance of our logic in discovering causal regularities. One of the important factors of this dataset is that there are many missing values/data points. Our algorithm is designed to work on datasets with missing data which makes this dataset a great example of this capability. Another important factor of this dataset is its limited number of datapoints. There are 307 data points (individual plants) and none of the 19 classes (different diseases) have more than 40 examples.

For this test, we only included things that we knew could be causes of anthracnose. The only predicates allowed were: temperature, precipitation, hail, and treatment type. This resulted in a dataset with 307 rows (plants) and 14 columns (event types)

6.3.2 What we Expect: We tested our algorithm by looking for things that prevent anthracnose, which is a fungal soybean disease. This test represents a scenario where a soybean farmer wants to prevent it in their crop, or just wants to know what can cause anthracnose. As you will see, these causes can be discovered by telling our algorithm to find the causes of anthracnose, and to find the causes of the lack of anthracnose. Since it is well known by soybean farmers that anthracnose is known to occur during warm, wet, and humid conditions, we expect our system to signal this fact.

6.3.3 What the Algorithm Actually Identifies: When the algorithm was run using anthracnose as the chosen effect, the algorithm returned no proven conditions. This means that there are no combinations of factors that guarantee anthracnose. This could be somewhat useful to a farmer or soybean researcher, but the more interesting findings are when the algorithm was run using the lack of anthracnose as the chosen effect. When the lack of anthracnose was the chosen effect, 154,295 conditions were tested, resulting in the following proven conditions:

[('temp_?',),
('precip_1',),
('temp_0',),
('precip_0',),
('~precip_2',),
('precip_?',),
('treatment_?',),
('temp_1', 'hail'),
('~temp_1', '~temp_2'),
('~temp_2', 'hail'),
('~temp_1', '~hail'),
('temp_2', '~hail'),
('~treatment_0', '~treatment_1', '~treatment_2')]

Since many of the predicates were implemented with one-hot encoding (See diagram 1), some of these conjunctions should be ignored (an underscore in the attribute name ex: temp_1, signifies that one-hot encoding was used). Some of these should be ignored because they are an artifact of one-hot encoding. For example, if we know that temperature is not high and not normal, we know that it must either be low or unknown, and this logical entailment in itself says nothing about the data set itself.

The principal takeaways from the output are that anthracnose never occurs under the following conditions: there is normal or less than normal precipitation, the temperature is less than normal, the temperature is normal and there is no hail, the temperature is greater than normal and there is hail or when the treatment is unknown. The treatment being unknown is odd since one might assume the data collectors would have that information, but it is unlikely to provide any useful information for this so it will be ignored for the analysis. One thing that should be considered when analyzing these results is that anthracnose is known to occur during warm, wet, and humid conditions. This only bolsters our results, since our algorithm showed that low temperature prevents anthracnose, and it only occurs when there is greater than normal precipitation. Since this is already known, we have shown the algorithm's ability to discover causal regularities.

7 Time Complexity Analysis

7.1 Algorithm Analysis

The algorithm is made up of 3 main parts: reading the dataset, generating possible MINUS conditions, and verifying or

discarding all the possible MINUS conditions. Reading the dataset iterates over all predicates (P) and all data points (n). This gives a total time of O(nP).

When the algorithm generates the potential MINUS conditions, it does the following for every data point; prepares the dataset for analysis O(P), finds the set of all subsets from the data point's set of predicates O(2^P), and adds the members of this set to the set of all potential MINUS conditions O(2^P). Since there are n data points, generating MINUS conditions is O(n(P+2^P + 2^P)) which is equivalent to O(n2^P).

Establishing or rejecting each MINUS condition iterates over the set of all potential MINUS conditions, which can be as large as 2^P. For each of these potential MINUS conditions, we check that it is not a superset of any proven MINUS condition. Iterating over all proven MINUS conditions can be as large as O(2^P). Then each MINUS condition that survives this winnowing is checked against every datapoint. This takes O(nP^2) time for each potential condition it checks. The total time for this section is O(2^P(2^P + nP^2)) which is equivalent to O(2^{2P}). However, since there are often a much smaller number of verified MINUS conditions in a real dataset, in practice the upper bound is often O(n$P^2 2^P$).

Combining all these sections results in a final time complexity of O(2^{2P}). However, in practice this is closer to O(n$P^2 2^P$).

7.2 Theoretical Minimum Time Complexity

This section calculates the maximum possible set of proven MINUS conditions. If it is assumed that an algorithm takes O(1) time to compute and output each MINUS condition, this yields a theoretical lower bound for an algorithm that generates MINUS conditions. However, such an algorithm would be unrealistic, so this will only be used as a way to analyze the algorithm described in this paper.

By way of proof, there are $\frac{P}{P/2}$ sets in the largest set of MINUS conditions. Since each one of these sets is no larger than P-1, if it takes O(1) to generate and output each part of each MINUS condition, the total time complexity would be O(P$\frac{P}{P/2}$). P$\frac{P}{P/2}$can be expanded to $P(\frac{P!}{(P/2)!(P/2)!})$. Using Stirling's approximation, P! is equivalent to $\sqrt{2\pi P}(\frac{P}{e})^P$ as P approaches infinity. However, it is more accurate to find the upper and lower bound of P $\frac{P}{P/2}$ using the upper and lower bound of Stirling's approximation. For the upper bound, the numerator will use the upper bound of Stirling's approximation $eP^{P+1/2}e^{-P}$, and the denominator will use the lower bound of Stirling's approximation

$$\sqrt{2\pi}(\frac{P}{2})^{\frac{P+1}{2}}e^{-\frac{P}{2}}$$

This will maximize the approximation of P$\frac{P}{P//2}$. Substituting these upper and lower bound approximations yields

$$P\frac{eP^{P+1/2}e^{-P}}{(\sqrt{2\pi}(\frac{P}{2})^{\frac{P+1}{2}}e^{-\frac{P}{2}})^2}$$

Distributing the exponent in the denominator gives

$$P\frac{eP^{P+1/2}e^{-P}}{2\pi(\frac{P}{2})^{P+1}e^{-P}}$$

Canceling out e^{-P}, moving 2^{P+1} to the numerator and simplifying gives $\frac{e}{\pi}\sqrt{P}2^P$. Following the same process but with the lower bound of Stirling's approximation in the numerator and the upper bound in the denominator yields the lower bound for P$\frac{P}{P/2}$. This lower bound is

$$\frac{2\sqrt{2\pi}}{e^2}\sqrt{P}2^P$$

Since both the upper and lower bound are O($\sqrt{P}2^P$), the time complexity of this theoretical algorithm is O($\sqrt{P}2^P$).

As mentioned previously, this theoretical algorithm is unrealistic since it can find each MINUS condition in O(P) time, and it does not account for the number of data points. With this in mind, the algorithm described in this paper is mathematically O(2^{2P}), but in practice is closer to O(n$P^2 2^P$). Both algorithms take exponential time which shows that the algorithm described in this paper is within the same time complexity class as the theoretical minimum.

8 Conclusion

The identification of causal relationships in datasets can be very illuminating. There have been various approaches to the notion of causality, and through research and experimentation we have built upon these approaches to create a well-rounded algorithm for identifying causal relationships.

As shown in the results portion, we have proved that our algorithm successfully identifies singular and conjunctive conditions that serve as possible causes for a chosen event. The three datasets we have discussed exhibit the key features of why our algorithm is useful in finding these causal relations and also evidence of the accuracy of the algorithm that we have composed. The soybean dataset test illustrates the importance of finding causes for the inverse of an event and also illuminates the fact that our algorithm cannot distinguish between causes and symptoms of an effect. The algorithm finds correlations, but it is up to the user to only include things that would be causes and not symptoms, or the user would have to manually analyze the output and determine its likelihood of being a cause or symptom.

The causal regularities that the algorithm generates are reliably predictive because they are only produced if there is

certainty, they will hold for a given data set. This predictive capability arouses a comparison to machine learning, arguably the most popular method for an algorithmic approach to prediction. Two of the biggest problems with machine learning are the need for lots of data, and the inability to see why a trained model makes a decision. Our algorithm doesn't suffer from either of these problems, but still creates causal regularities that can be used to make accurate predictions. More data is always useful for better prediction for both machine learning and our algorithm, but our algorithm requires a much smaller amount to get meaningful results. Even with only 20 examples of plants with anthracnose (a much too small amount for any standard machine learning algorithm) we were able to find useful information about the causes of anthracnose, and how to prevent it with certainty.

Another advantage of this algorithm is its 'white box' character; that is, it is straightforward to reconstruct the results obtained. This is useful for a dataset like the heart disease dataset because knowing the conditions and conjunctions of conditions that may cause heart disease can help individuals seek help sooner and also take care of their health in order to prevent themselves from being diagnosed with heart disease. Because we know that the causal regularities we generate will always hold true for the given dataset, then if we are confident that the dataset is fully representative of the problem, we will only generate ironclad predictions that will almost certainly always hold true.

Other methods of prediction can be effective for large datasets that don't require human understanding, but our algorithm can use small datasets to create precise predictions that are easily interpretable. Outside of prediction, our algorithm is also able to discover unknown properties in these datasets, generating new knowledge that other approaches could not gain.

9 Future Work

This implementation is open to sundry developments. First, to be a habitable system, a more congenial user interface is advisable. In addition, a redesign of the basic search for MINUS conditions along the lines of the *A Priori* algorithm for frequent item sets in market basket analysis would be possible. This *A Priori* approach would cut down on mathematical time complexity by a power of 2. However, in practice, this decrease in time complexity would be significantly lower. Beyond these matters of performance and cosmetic ease of use, its functionality might be augmented in several ways:

1. First, at present the algorithm detects MINUS conditions, i.e., conjunctions of conditions which are sufficient for the effect. It also assembles all such conditions such that, in the input data set, their disjunction is collectively necessary, i.e., one such conjunction must be present for the effect to occur. The algorithm does not explore the relative importance of each individual conjunct. Although most theories of causal regularity do not provide much guidance, we sense the possibility of isolating events of particular importance through abstraction; that is, two attributes may be instances of the same abstraction, and might be consolidated into a single event type. For example, if the data set has three attributes 'is colorless,' 'is green,' 'is blue,' and it turns out that the conjunction of 'is green' and 'is blue' figures in a MINUS condition, it might be abstracted to the simpler 'is colored'.
2. Metrics analogous support and confidence such as are found in itemset mining are appropriate and potentially useful. The algorithm will reliably organize the data into the conjunction of MINUS conditions, but they may apply only to a few data points and hence have little predictive force. Measure so confidence might be appropriate if we relax the stipulation that MINUS conditions are identified only if they have complete predictive power.
3. The existence of unknown values for certain predicates and objects, arouses the possibility of suggesting experimental designs. For example, discerning that a known positive or negative value for a given attribute would establish an additional causal hypothesis, could be useful in exploring the phenomenon summarized in the given data set. Also, discovering that the elimination of an attribute with many unknown values might lead to more definitive results could lead to greater insight into a data set.

References

[1] Michale Baumgartner, "Regularity Theories Reassessed," *Philosophical*, 36:327-354, 2008.

[2] Micheal Baumgartner, "Uncovering Deterministic Causal Structure: A Boolean Approach," *Synthese*, 170(1):71-96, 2009.

[3] Mathieu Beirlan, Bert Leuridan, and Frederik Van De Putts, "A Logic for the Discovery of Deterministic Causal Regularities," *Synthese*, 195:367-399, 2008.

[4] G. Graßhoff and M. May, "Causal Regularities, W. Spohn, M. Ledwig, and M. Esfield (Eds.) *Current Issues in Causatio*n, Paderborn: Mentis, pp. 85-214, 2001.

[5] J. L. Mackie, "Causes and Conditions," American Philosophical Quarterly, American Philosophical Quarterly, 2(4):245-264, Oct. 1964.

[6] J. L. Mackie, *The Cement of the Universe: A Study of Causation*, Oxford: Clarendon Press, 1984.

[7] J. S. Mill, *A System of Logic*, London: John W. Parker, 1843.

[8] Judea Perl, *Causality. Models, Reasoning and Inference*, Cambridge University Press, Cambridge, 2009.

Thomas Bidinger (photo not available) recently graduated with baccalaureate degrees in Computer Science from Western Washington University.

Hannah Buzard (photo not available) recently graduated with baccalaureate degrees in Computer Science from Western Washington University.

James Hearne (photo not available) is a Professor of Computer Science at Western Washington University, where his research is focused on the application of data mining and machine learning techniques to the interpretation of ancient historical records.

Amber Meinke (photo not available) recently graduated with baccalaureate degrees in Computer Science from Western Washington University.

Steven Tanner (photo not available) recently graduated with baccalaureate degrees in Computer Science from Western Washington University.

Integration of Multimodal Inputs and Interaction Interfaces for Generating Reliable Human-Robot Collaborative Task Configurations

Shuvo Kumar Paul*

University of Nevada, Reno, NV, USA

Pourya Hoseini, Arjun Vettath Gopinath, Mircea Nicolescu, Monica Nicolescu†

University of Nevada, Reno, NV, USA

Abstract

As robots become more ubiquitous in our daily life, designing natural, easy to use, and meaningful interaction interfaces relevant to robotic tasks is vitally important as not only it can enhance user experience, but also can increase task reliability by proving supplementary information. This paper presents a flexible framework that integrates two natural interaction interfaces: speech, and pointing gesture with the sensor input streams to generate reliable task configurations for human-robot collaborative environment. The proposed framework takes the RGB image as input to detect the objects present in the scene and to recognize the pointing gestures, and it computes the corresponding pointing direction in the 2D image frame to infer the target object in the scene. At the same time, verbal instruction is received from the audio input which is then converted to text to either be fed into the proposed neural model or to compare against predefined grammar rules to extract relevant task parameters. All this information is used to resolve any missing or ambiguous task parameters. Structured task configurations are formed for the desired human-robot collaborative tasks. The proposed framework shows very promising results in integrating the relevant task parameters for the intended robotic tasks in different real-world interaction scenarios.

Key Words: Interaction interface; gesture recognition; multimodal inputs; input integration; robotics; human-robot interaction; natural language processing.

1 Introduction

Recent technological advances in automation, engineering, and artificial intelligence have provided the impetus for the rapidly accelerating robotics revolution. While in the past few decades the number of industrial robots skyrocketed, only recently robots are becoming more and more inclusive in our daily life. The rapid advances in AI and robotics have significantly shifted the focus of robotics research from industrial robotics to service robots; these robots lend a helping hand for tasks like cooking, cleaning, assistance, companionship, education, and so on. In contrast to the industrial robots, which repeatedly perform a specific task, a service robot is expected to interact with humans while performing tasks, and an ideal interaction should replicate a human-to-human interaction.

Designing interaction interfaces that are more intuitive and instinctive are prerequisites for ensuring the ease of use and inclusion of robots in our daily life. However, to sustain this inclusion, robots would need to build and maintain the trust of the user, particularly the trust that the robot can reliably perform a designated task. This warrants Human Robot Interaction (HRI) framework that not only can establish a natural interaction interface, but also can supplement the robotic task execution with additional data to make it more reliable.

Robotic entities are set to become an essential part of modern society and have the potential to shape our social experience. However, the inclusion of robots in our daily life will be dictated by one key factor: our trust in robots, specifically, the confidence of the human user that a robotic agent can accurately perform a task. To build and maintain this trust, it needs to be made sure that the robots can consistently execute the tasks in a proper manner.

Proper execution of robotic tasks requires instructions containing a set of parameters that defines a task configuration. We have identified two essential properties of a **complete** task configuration:

1. The task configuration needs to have all the relevant task parameter information for executing an intended task. Depending on the task, the task configurations can have different task parameters e.g. navigational task may only require the direction, while an assisting task may involve

*shuvo.k.paul@nevada.unr.edu

†hoseini,avettathgopinath@nevada.unr.edu;mircea,monica@unr.edu

knowledge of object(s), their attributes and locations in the environment, order of task information, etc. The relevant task parameters can be extracted from the robot sensors and filtered to form a set of structured instructions that can be correctly interpreted by the robotic entity.

2. The task configurations should be reliable enough for intended task execution. The robot may follow the instruction accordingly, but the instruction itself may contain erroneous task parameters; for instance, the robot may start moving toward its left, while the intended instruction was to turn left. This can result either from the noisy sensory data or ambiguities during task parameters extraction. To address this, the instructions need to be cross-validated to ascertain the intent of the user which can be done by integrating sensory information with different human robot interaction interfaces.

The first property is sufficient for executing a robotic task, however, the second property provides more assurance for the validity of the task configurations. Subsequently, if required task parameters can not be inferred from sensor input streams, then the interaction interfaces can be used to determine and/or verify the task parameters and vice versa. Although, achieving the second property may not be viable for every task configuration, verifying instructions from multiple input streams and interaction interfaces should definitely be one of the objectives in collaborative HRI design as it will help generate more reliable task configurations. More reliable task configuration would aid in proper task completion, which would help build more trust in robots as well.

Natural human interactions mostly involve gestures, speech, and facial expressions. Amongst these interaction interfaces, gestures and particularly pointing gestures are probably the most natural for humans and can serve as an effective device to convey a simple message or a command to the robot. Unlike speech, pointing gestures are not suitable for conveying sophisticated information; however, it provides with a more natural interface for simpler instructions that can be relayed even in noisy environments. Moreover, it provides the possibility of specifying objects and their locations intuitively and can be used as simple but meaningful commands. In addition, distinct human gestures represent specific information that can be used to convey the general intent of the user. This inferred intent information then can be compared or matched to predefined gesture configuration to provide additional information or appropriate command for the robot to execute certain tasks.

At the same time, speech can be used as a natural medium to express intricate commands which can then be used to effectively apply modern Natural Language Processing (NLP) techniques to parse and extract language information. Proper utilization of natural language can permit for a more intrinsic and faster dialogue between human and the robot. However, natural language communication needs to be translated into a formal language which the robot can process and make a decision upon. Subsequently, the robot needs to formulate a structured message for the successive communication which needs to be transformed into natural language to allow the user for easier comprehension.

Gestures information can simultaneously be integrated with verbal communication (speech recognition) to provide auxiliary information to further disambiguate natural language commands.

In our work, we focused on integrating two interaction modalities: 1) pointing gesture that can be used to direct the attention of the interacting robot toward an object or a certain location in the scene, and 2) verbal commands which are translated into simpler formal languages that are more interpretable for the robot; both of these are natural interaction interface for humans. Additional data containing the information of detected objects in the scene is also passed to the system to find and locate the **Object of Interest (OOI)**. **OOI** is the object that is requested by the human user to be manipulated by the robotic entity.

We propose a simple and reliable HRI framework that extracts a set of information from verbal instructions retrieved from the audio input, detects pointing gestures, estimates the general pointing direction and the object being pointed at from individual RGB frames, and finally, assigns them to the appropriate task parameters which then can be used to formulate structured instructions. We have focused primarily on simpler but more common collaborative task instructions targeting scenarios where a user provides navigational instructions e.g. "go to your right", "go there", etc. or commands that require object manipulation e.g. "give me the bowl", "bring that red book", etc. The task configuration has the following parameters: $\underline{a}$: action, $\underline{o}$: general object, $\underline{r}$: object attribute, $\underline{p}$: position of the object in the scene, $\underline{d}$: general pointed direction, $\underline{o_p}$: estimated pointed object. These extracted parameters can be used to generate a formal task description targeting a specified goal. Our approach relies on Google's speech to text API and the skeletal joint points extracted by AlphaPose [5, 15] from the RGB image to infer gesture.

The main contributions of this paper are of three folds:

1. We present a gesture recognition system that estimates whether the user is performing a pointing gesture and the pointing direction.
2. We leveraged a verbal command comprehension system to extract certain information from a user command relevant to specific robotic tasks by a) deep multi-task learning model and b) matching to pre-defined language patterns.
3. We have implemented an object detection and pose estimation system using template matching technique suitable for fast prototyping and demonstrated how the pointing gesture framework coupled with the extracted information from verbal command can be used to generate a list of task configurations corresponding to a sequence of tasks.

This paper is outlined as follows: in the next section, we provide a brief overview of previous work on gesture recognition techniques and natural language understanding in

HRI design. Next, we describe the methodology of our work in detail. The following chapters include our evaluation including experimental results and observations. Finally, we conclude this paper by summarizing our work.

2 Literature Review

A large body of work has been published in the area of HRI that focuses on pointing gestures and natural language understanding. In the following sub-sections, some of the previous works have been discussed.

2.1 Pointing Gesture Recognition

Early pointing gesture interfaces were built with the help of wearable devices e.g. glove-based devices [25, 8]; Dipietro et al. [2] surveyed Data-GLove like systems and their application for gesture recognition. Kahn et al. [10, 11] introduced Perseus architecture which operated on several feature maps (intensity, edge, motion, disparity, color) to locate pointed objects by interpreting the pointing gestures. With the continuing advances in computer vision, gesture recognition research shifted more toward vision-based methods. Kadobayashi et al. presented VisTA-Walk, a gesture interpreter which could only infer left or right; it uses the output recognition result of Pfinder [32] which employed a multiclass statistical color and shape model to derive a 2D representation of head and hands from a wide range of viewing conditions. With the introduction of stereo cameras, multi-cameras, time-of-flight (TOF) cameras, or depth cameras like Kinect, Intel RealSense, etc. researchers were able to come up with different approaches for solving pointing gesture detection. [30, 12] used multi-camera setup while [3] used TOF camera to segment body, localize forearm and elbow for gesture detection, and subsequently used Gaussian Process Regression for pointing direction estimation.

Different Hidden Markov Model (HMM) have been used to detect pointing gestures. Wilson et al. [31] proposed parametric HMM for recognition, representation, and interpretation of parameterized gestures, such as pointing gestures. Jojic et al. [9] used only the dense disparity maps for gesture detection, while Nickel et al. [20] calculated the dense disparity maps to track the positions of a person's face and hands together with an HMM-based approach for detecting pointing gestures. Park et al. [21] applied Cascade HMM and particle filters but depended on a large number of HMM states for accurate gesture recognition which required large amounts of training data and thus, incurred higher processing time.

Richarz et al. [27] used Gabor wavelets to extract features and multilayer perceptron to approximate pointing direction estimator, but it was sensitive to pose variations. Pateraki et al. [22] exploited the prior information of the location of possible pointed targets and used the Dempster–Shafer theory of evidence to fuse the estimated head pose and hand pointing orientation information to locate the pointed target.

Rautaray et al [26] extensively reviewed hand gesture recognition and pointed out the well known limitations of the leading technologies related to the field.

In our work, we used the estimated (pixel) location of the forearm joints (elbow and wrist) of the user to 1) determine whether the person is performing the pointing gestures and 2) infer the general direction the user is pointing e.g. left, right, straight and estimate the pointing direction by computing the line that goes through the arm joints.

2.2 Natural Language Understanding in HRI

Natural language based interaction has been explored in various human-robot interaction tasks that include instructing the robot with direction for navigation, commanding for performing certain robotic tasks, specifying the object to manipulate, etc. Natural language understanding is also used as a modality along with other sensory information like vision to disambiguate human instructions or the scene configuration in general.

Kollar et al. [13] presented a system that infers the probable path for an agent by taking the environmental geometry and the detected objects as inputs along with the extracted sequence of spatial description clauses from the linguistic information. Matuszek et al. [18] investigated statistical machine translation techniques to follow natural language route instructions within a tractable manner. Macmohan et al. [17] introduced MARCO, an agent that infers implicit actions from knowledge of linguistic conditional phrases and spatial action information along with environmental configuration. This method performs the explicit, implicit actions required to reach the instructed state, and subsequently executes exploratory actions to learn about the environment. In [29] an approach was introduced to automatically generate a probabilistic graphical model with respect to the hierarchical and compositional semantic structure of natural language navigation or mobile manipulation commands.

Dzifcak et al. [4] proposed an integrated robotic architecture that translates natural language instructions incrementally and simultaneously generating logical goal representation and action language, which can be further analyzed to measure the achievability of the goal as well as to create new action scripts targeting specified goals. Kuo et al. [14] demonstrated that the combination of a hierarchical recurrent network with a sampling-based planner can be utilized to generate a model that learns to understand a sequence of natural language commands in a continuous configuration space. The use of spatial relationships to establish natural communication mechanism between humans and robots was investigated in [28]; a multimodal robotic interface comprising of linguistic spatial descriptions and other spatial information extracted from an evidence grid map was used to show how this information can be used in a natural, human-robot dialog. [1] described a robotic architecture featuring a planner that utilized discovered information by learning the pre and post conditions of previously unknown action sequences from natural language

construction.

We have leveraged the recent natural language processing methods in our work to retrieve text from speech, generate grammatical patterns to match, and extract relevant command information from the user.

3 Methodology

The system receives gesture information and verbal communication from an RGB image, and an audio input stream respectively. The speech in audio is converted to text to further extract the action instruction and the object information corresponding to the robotic tasks. RGB image is used for pointing gesture recognition, pointed direction estimation along with object detection, and pointed object prediction. The overview of the system architecture is illustrated in the Figure 1; the green boxes indicate the extracted task parameters.

3.1 Information Extraction from Verbal Commands

In the course of a typical human-to-human collaborative interaction, the instructions that the participants interchange usually consists of a set of particular information; this includes the action to be performed, the object of interest, direction information for navigation, location of interest in the scene, etc. Furthermore, humans generally describe an object in terms of a general color, pattern, shape, size, and the relative position to disambiguate [24], e.g., "bring that red shirt", "The book on the left", "take the small box", etc. This information defines certain parameters of a task. We have developed two techniques: 1) neural network model, 2) natural language pattern matching, to extract the task parameters from verbal commands. These two approaches are outlined in the following sub sections.

3.1.1 Neural Network Model. For our work, we generated a dataset for collaborative robotic commands. Subsequently, we considered 8 different architectures for training our model and found single layer Bi-directional Long Short Term Memory (Bi-LSTM) based model to be optimum. As our goal was to extract multiple task parameters from the verbal commands, we formulated deep multi-task learning model. Multi task learning (MTL) is a sub-division of machine learning, where multiple tasks are jointly learned by a shared model. Deep multi task learning tries to produce a generalized representation that are powerful enough to be shared across multiple tasks; here, each task denotes a multi-class classification.

Dataset: We generated a dataset of commands where each of the commands contains action information, and information about one or more of the following three things: object name, object color, and object size. The dataset contains 60769 samples, each of which has four labels.

Model Architecture: The model contains three neural layers: an embedding layer, a Bi-LSTM layer, and a fully connected layer. Let's assume the vocabulary length of the dataset is $\mathtt{V}$, then every word is represented with a one-hot encoding of size $W \in \mathbb{R}^{1\times \mathtt{V}}$. The input sequences or sentences each contains n elements or words. These inputs are fed into an embedding layer $\mathscr{E}$.

An embedding is a mapping of a discrete or categorical values to a vector of continuous numbers. A neural netwrok embedding provides a low-dimensional, learned continuous vector representations of these discrete variables. The key advantage of neural network embeddings is that they can reduce the dimensionality of categorical variables to represent them in the transformed space. $\mathscr{E} \in \mathbb{R}^{\mathtt{V}\times \mathtt{d}}$, where $\mathtt{d} << \mathtt{V}$ denotes the lower dimensional embedding vector; this lower dimensional vector is then passed to the Bi-LSTM layer.

LSTMs [7] are a particular form of Recurrent Neural Network (RNN). LSTM (or Bi-LSTM) are ideal for sequential data such as text, speech, video, audio, etc. Our key motivation for choosing Bi-LSTM is that it can make use of both the past and future context information of a sentence, and can learn long-term temporal activities as well as avoid exploding or vanishing gradient that the traditional RNN suffers from during the back propagation optimization. In Figure 2, f_i and b_i denote forward and backward LSTM respectively.

The output of the Bi-LSTM cells are concatenated and fed into the four fully connected (FCN) layers. Finally, the output of the FCN layers goes through softmax activation to classify four task parameters. For each classifier, we measured the Cross Entropy loss $\mathscr{L}_c$ and used the mean of these losses $\mathscr{L}_m = \frac{1}{4}\Sigma_{c=1}^{4}\mathscr{L}_c$ to update our model.

3.1.2 Natural Language Pattern Matching. A set of language patterns can provide sufficient information about the parameters of collaborative interaction while excluding other verbal communications that may not contain a command. The language patterns are arranged in Table 1. These patterns are represented in terms of regular expressions with different language elements as terms. The angled brackets(<>) encloses each term. The terms with capital letters indicate different language elements, while the exact words are enclosed within quotes. An array of words, surrounded by square brackets e.g. [”right”, ”left”, ”front”, ”back”], specifies only one of them needs to be present. The symbols after each term e.g. +, ?, * are the quantifiers that represent how many times the preceding term needs to be matched; + indicates the term needs to be matched 1 or more times, * means 0 or more, ? means 0 or 1, and if there are no quantifiers then the preceding term needs to be matched exactly once.

The **Action** pattern extracts the general task needed to be executed; simultaneously, it can distinguish between a traditional instructive command and a non-instructive verbal communication. The **Object** pattern is almost similar to **Action** pattern except for that it needs to match with a *NOUN* at the end which would contain the name of the object. The **Attribute** pattern identifies certain visual characteristics of the object and

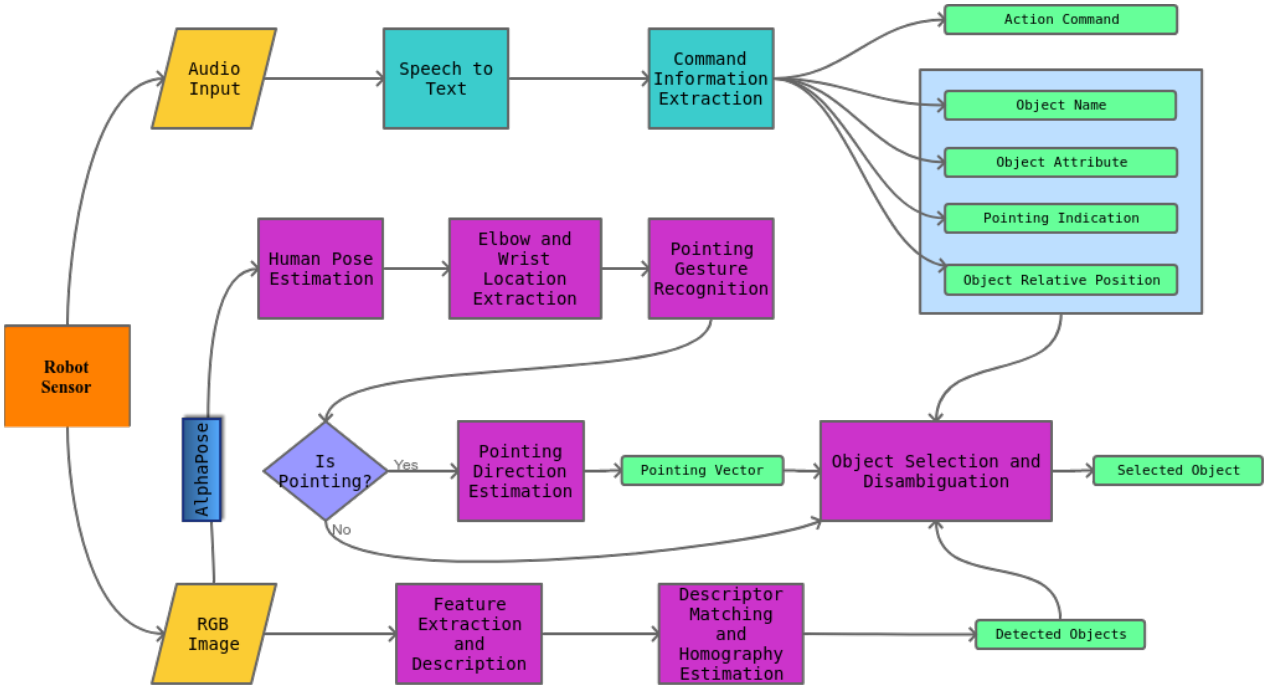


Figure 1: System architecture

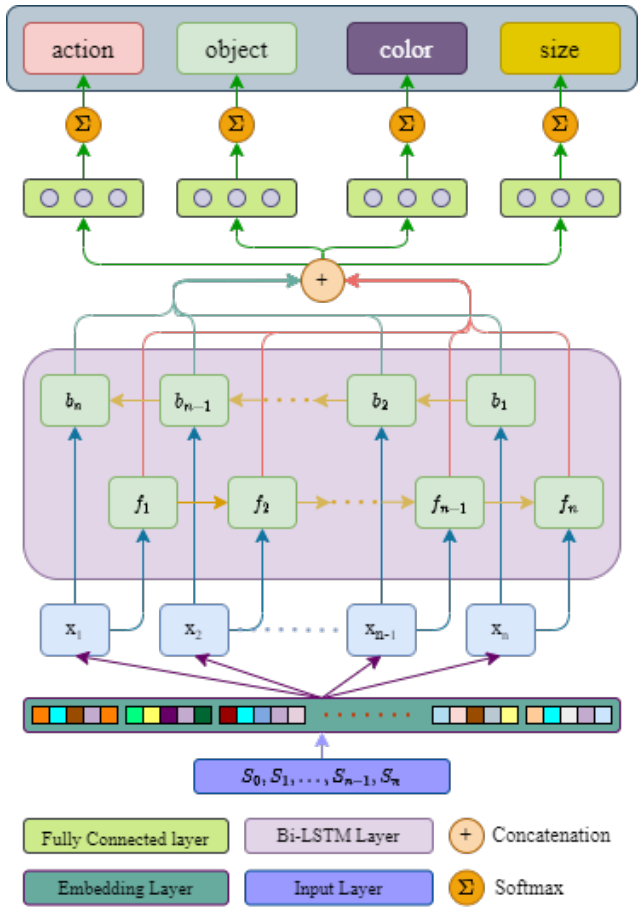


Figure 2: NN model for parameter extraction from verbal commands.

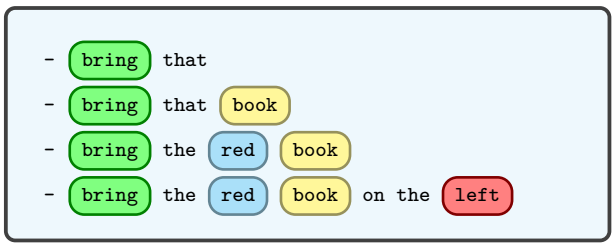


Figure 3: Green box represents the task action; red box indicates the location of the object in the scene; yellow and blue boxes specify the object of interest and the corresponding attributes

the **Position** pattern determines the general location, both of which can be utilized as identifiers/specifiers to disambiguate objects that fall into similar categories. Figure 3 presents a set of sample instructions iterated with additional information and illustrates the retrieved relevant information as well.

3.2 Pointing Gesture Recognition

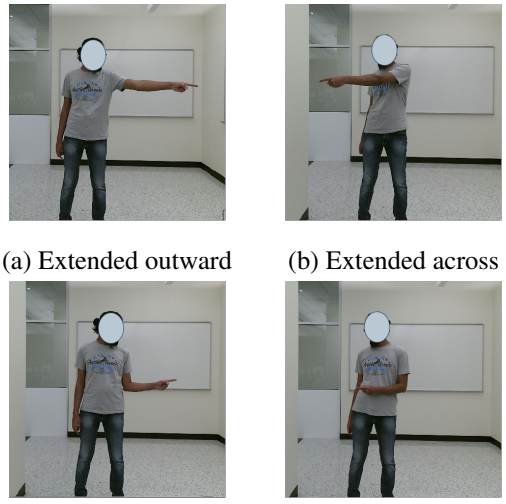
AlphaPose [5] was used to extract the skeletal joint locations to predict the pointing gestures and the general pointing direction. For simplicity, we assumed the user is using one hand at a time for pointing. Park et al. [21] categorized the pointing gestures into large and small pointing gestures, which we labeled them as extended (Figure 4(a, b)) and bent (Figure 4(c, d)) arm gestures. Additionally, for the pointing gesture, the forearm's relative direction with respect to the body can be generalized in three categories: across (Figure 4(b, d)), outward (Figure 4(a, c)), and straight (Figure 5(b)).

For across and outward pointing gestures, we can simply measure the angle θ_a (Figure 5(a)) of each forearm with respect to a vertical line and compare it to some smaller angle threshold θ_t to determine whether the user is performing the pointing gesture or not. In other words, if the user is not pointing (Figure 5(b)) then the forearm would produce a smaller angle compared to when the user is pointing. However, if the user points straight with respect to the camera (robot's vision) (Figure 5(c)), the angle would be close to 0 and to address this, we measured the ratio of the lengths of the forearms ρ_a; intuitively, if the user is not pointing, the lengths of the detected forearms should be virtually the same (Figure 5(c)), but if one of the forearm's length is significantly shorter than the other, it can be assumed that the user is pointing straight (or its proximity) toward the camera using that corresponding arm (Figure 5(b)). Furthermore, the general direction d of pointing was determined from the relative position of the wrist and the elbow of the pointing arm to supplement navigational command.

3.2.1 θ_a Calculation from Wrist and Elbow Location. From the extracted skeletal joints' locations, only the following joints were needed: left elbow, left wrist, right elbow, and right wrist. This means even if some body part is occluded, our approach should still work as long as pointing hand's joints are detected. Let us define the skeletal joint coordinate of the elbow as (x_1, y_1), the wrist as (x_2, y_2); the pointing 2D vector centered at the origin can be defined as $\vec{a} = (x_2 - x_1, y_2 - y_1$ and the vertical vector to compare with is set to $\vec{v} = (0, 1)$. The pointing angle θ_a is measured using equation 1.

$$\theta_a = \cos^{-1} \frac{\vec{a} \cdot \vec{v}}{|a||v|} \tag{1}$$

If $\theta_a > \theta_t$, then we consider the corresponding forearm is performing the pointing gesture and subsequently, compare the x coordinate of the wrist and the elbow to further detect the general pointing direction towards the left or right of the scene (across or away with respect to the body). Next, the ratio of the length of the forearms $\rho_a = \frac{\text{Length of the arm of interest}}{\text{Length of the other arm}}$ is compared against a ratio ρ_t to further decide whether the user is pointing straight. We set 0.8 as the value of ρ_t and 15° for θ_t.



(a) Extended outward (b) Extended across

(c) Bent outward (d) Bent across

Figure 4: Gesture categories

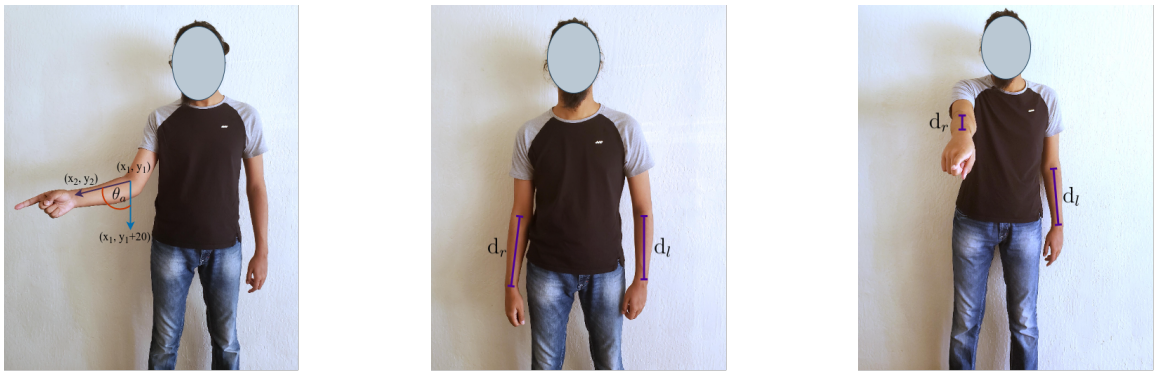
3.3 Object Detection

Our object detection module consists of two phases: (i) feature extraction and matching, and (ii) homography estimation and perspective transformation. In the following sections, we will focus on the detailed description of the aforementioned steps.

3.3.1 Feature Extraction and Matching. Our object detection starts with extracting features from the images of the planar objects and then matching them with the features found in the images acquired from the camera. Image features are patterns in images based on which we can describe the image. A feature detecting algorithm takes an image and returns the locations of these patterns - they can be edges, corners or interest points, blobs or regions of interest points, ridges, etc. This feature information then needs to be transformed into a vector

Table 1: Grammatical patterns for information extraction

Information Type	Grammatical Pattern	Example
Action	<VERB><"me">?<DET‡ >?<["this", "that"]>?<ADJ§>*<>?	bring me that; give me that book;bring it here
Object	<VERB><"me">?<DET>?<["this", "that"]>?<ADJ>*<NOUN>	take the red cup; bring me that dress
Attribute	<ADJ>+<NOUN>	red dress; large blue bowl
Position	<["right", "left", "front", "back"]>	the box on the left



(a) (b) (c)

Figure 5: (a) Generated angle θ_a , (b) length of forearms $d_l, d_r$when not pointing, and (c) straight

space using a feature descriptor, so that it gives us the possibility to execute numerical operations on them. A feature descriptor encodes these patterns into a series of numerical values that can be used to match, compare, and differentiate one feature to another; for example, we can use these feature vectors to find the similarities in different images to detect objects as well as distinguish each object in the scene. Ideally, this information should be invariant to image transformations e.g. change in illumination of the scene, different degrees of image blur and compression, variance in viewpoint, etc. However, every feature detector and descriptor is unique and can be tolerant to certain image transformations and to certain degrees. We selected SIFT [16] as both the feature detector and descriptor for our work as it provides reliable detection with adequate speed as reported in [23].

Once the features are extracted and transformed into vectors, we compare the features to determine the presence of an object in the scene. Usually, the Nearest Neighbor algorithm is used to find matches, however, finding the nearest neighbor matches within high dimensional data is computationally expensive, and with more objects introduced it can affect the process of updating the pose in real-time. To counter this issue to some extent, we used the FLANN [19] implementation of KD-Tree Nearest Neighbor Search, which is an approximation of the K-Nearest Neighbor algorithm that is optimized for high dimensional features. Finally, if we have more than ten matches, we presume the object is present in the scene.

‡DET refers to the determinant i.e. "the"

§ADJ refers to adjective part of speech e.g. small, red, etc.

3.3.2 Homography Estimation and Perspective Transformation. A homography is an invertible mapping of points and lines on the projective plane that describes a 2D planar projective transformation (Figure 6) that can be estimated from a given pair of images. In simple terms, a homography is a matrix that maps a set of points in one image to the corresponding set of points in another image. We can use a homography matrix **H** to find the corresponding points using equation 2 and 3, which defines the relation of projected point $(x^{'}, y^{'})$ (Figure 6) on the rotated plane to the reference point (x, y).

A 2D point (x, y) in an image can be represented as a 3D vector $(x, y, 1)$ which is called the homogeneous representation of a point that lies on the reference plane or image of the planar object. In equation (2), **H** represents the homography matrix and $[x\ y\ 1]^T$ is the homogeneous representation of the reference point (x, y) and we can use the values of a, b, c to estimate the projected point $(x^{'}, y^{'})$ in equation (3).

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

$$\begin{cases} x^{'} = \dfrac{a}{c} \\ y^{'} = \dfrac{b}{c} \end{cases} \tag{3}$$

We estimate the homography using the matches found from the nearest neighbor search as input; often these matches can have completely false correspondences, meaning they

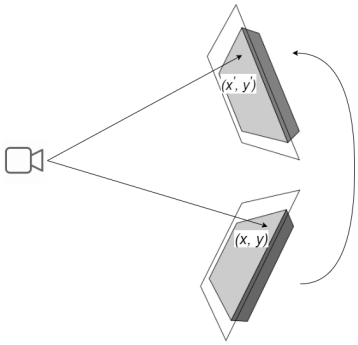


Figure 6: Object in different orientations from the camera

don't correspond to the same real-world feature at all which can be a problem in estimating the homography. So, we chose RANSAC [6] to robustly estimate the homography by considering only inlier matches as it tries to estimate the underlying model parameters and detect outliers by generating candidate solutions through random sampling using a minimum number of observations.

While the other techniques use as much data as possible to find the model parameters and then pruning the outliers, RANSAC uses the smallest set of data point possible to estimate the model, thus making it faster and more efficient than the conventional solutions. This estimated homography can also be effectively used for the planar pose estimation of textured objects [23].

3.4 OOI Estimation from Pointing Gesture

For each detected object, the bounding box can be defined as a list of four 2D line segments $BB = [s_1, s_2, s_3, s_4]$; s_i is defined by the following parametric equation:

$$s_i = (a_i, tb_i) = \left(V_i, \begin{cases} t(V_{i+1} - V_i) & \text{if } i < 4, \\ t(V_1 - V_i) & \text{else} \end{cases} \right) \tag{4}$$

where, V_i represents the $i^{th} (1 \leq i \leq 4)$ vertex of the quadrangle bounding box, $0 \leq t_i \leq 1$, and the value of t_i determines the location of a point on the segment; if $t_i = 0$ then it's the initial point and if it's equal to 1 then it's the final point in the segment (Figure 7). Additionally, the center of each detected object is computed by taking the average of the four vertices.

Similarly to equation 4, we can estimate the pointing direction by computing a 2D line from the pixel location of the arm joints (equation 5).

$$l_p = ((x_1, y_1), t(x_2 - x_1, y_2 - y_1)) = (\mathrm{a_p}, \mathrm{tb_p}) \tag{5}$$

where, l_p denotes the pointing direction in the image frame, $(x_1, y_1), (x_2, y_2)$ corresponds to the 2D pixel locations of the elbow and the wrist respectfully, and $-\infty < t < +\infty$.

For each s_i we can solve for t using equation 6 and find the intersecting point $p_i = a_i + t_i b_i$. Next, the distance from

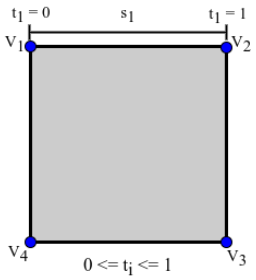


Figure 7: Visualization of the parametric equation of a segment

the object center to each corresponding intersecting point is measured and the minimum distance δ is computed and specified as the object distance δ. Algorithm 1 lists the steps for computing minimum distance δ for each detected object. Object with the least δ is estimated to be the pointed object.

$$t_i = (\mathrm{a_p} - a_i) \times \frac{\mathrm{b_p}}{b_i \times \mathrm{b_p}} \tag{6}$$

Algorithm 1: Minimum distance computation given 2D pointed vector and object boundary vertices

MinimumObjectDistance (l_p, V);
Input : l_p is the 2D pointing vector
V is a list of vertices representing the bounding box
Output: δ least distance from the object center
/* C is the center of the object */
$C = \frac{1}{4} \sum_{i=1}^{4} V_i$
$\delta = null$
for $i = 1$ *to* 4 **do**
  $s_i \leftarrow$ equation 4
  $t_i \leftarrow$ equation 6
  $p_i \leftarrow a_i + tb_i$
  $d = ||p_i - C||$
  if $\delta == null$ *or* $d < \delta$ **then**
    $\delta = d$
  end if
end for
return δ

Algorithm 2 describes the steps for extracting relevant task parameters.

4 Experimental Results

We set up experiments that involved the participants performing a specific pointing gesture within a predefined scenario. The scene contained three objects: two books, and a cheez-it box, and the user could point at one object at a given time. For instance, in one of the scenarios the user was

Algorithm 2: Task parameters extraction

ExtractTaskParameters (l_p, O);
Input : $\mathscr{I}$: RGB image
$\mathscr{C}$: verbal command
Output: Parameters of a robotic task
$[\mathrm{a}, \mathrm{o_g}, \mathrm{r}, \mathrm{p}] \leftarrow$ extracted from $\mathscr{C}$ using the language patterns arranged in Table 1
$[\mathrm{O}, \mathrm{BB}] \leftarrow$ detected object name and the corresponding estimated boundary from image $\mathscr{I}$
$J_l, J_r \leftarrow$ Extract elbow and wrist joints location of left and right forearm using AlphaPose
Estimate if the user is pointing and the pointing arm from J_l, J_r according to section 3.2
if *user is pointing* **then**
  $l_p \leftarrow$ compute the 2D pointing vector as described in section 3.2.1
  $d \leftarrow$ estimated general pointing direction
  /* object with minimum distance is the estimated pointed object $\mathrm{o_p}$ */
  $\mathrm{o_p} = \arg\min_V \{(l_p, V) | LeastObjectDistance(l_p, V), V \in BB\}$
  return $[\mathrm{a}, \mathrm{o_g}, \mathrm{r}, \mathrm{p}, \mathrm{o_s}, d]$
else
  return $[\mathrm{a}, \mathrm{o_g}, \mathrm{r}, \mathrm{p}]$

instructed to point at the leftmost object by extending their right hand; therefore, for this data sample, we know the user was performing a pointing gesture using their right hand, pointing to their left, and pointing at the rightmost object (leftmost from user's perspective). This information was set as the ground truth for quantitative evaluation. The participants were positioned at the center of the image frame and were instructed to point at different parts of the scene. The pointing direction was labeled as either away, across, or straight for the operating/directing hand and "not pointing" for the other. These experiments were carried out where the user was standing at 1.22, 2.44, 3.66, and 4.88 meters distance from the camera.

As our work depends on several modules, we have decoupled them to evaluate how each of the modules performs. These modules include extracting task parameters from verbal commands, detecting the operating hand, estimating the pointing direction, and predicting the object of interest. Finally, we have presented the final result in terms of extracted task parameters in a tabular form.

The system receives the verbal command and extracts the parameters using the NLP techniques described in section 3.1.1 and 3.1.2. We examined RNN and LSTM based model each with 4 different variants: bidirectional and non bidirectional; 1 and 2 layers. We found the single layer Bi-LSTM network to be superior in performance. Figure 8 illustrates the superior performance of the Bi-LSTM network compared to other neural network architectures in terms of validation accuracy.

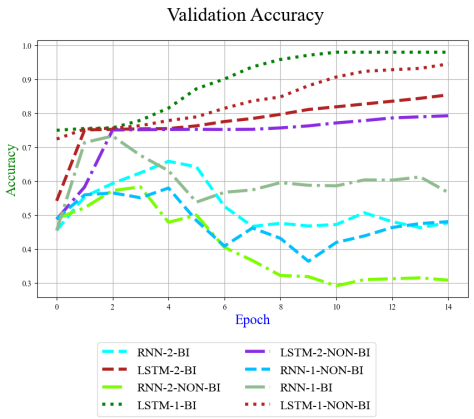


Figure 8: Validation accuracy

These parameters are stored so that each task can be executed sequentially. Table 2 arranges different verbal commands received from the user and the corresponding extracted task parameters; if no matches found, the corresponding parameters are set to ***None***. Each command initiates a task and is stored according to the order of task initiation (Table 3).

For each reliable frame, we compared the prediction with the label and measured the accuracy, precision, and recall. For a sample frame that has a label of "Right hand: pointing; Left hand: not pointing", if the prediction is "Right hand: pointing" then the sample is labeled as True Positive, else False Negative, if the prediction is "Left hand: pointing" then the sample is labeled as False Positive, else True Negative. Table 4 tabulates the accuracy, precision, and recall for varying distances. Figure 11 illustrates the system output for different pointing scenarios.

Next, we experimented on scenarios where multiple objects were placed on top of the table, each of which had predefined attributes (Figure 9). Additionally, the participants were instructed to point at a specific object. The system takes this information along with the natural language instruction to devise the task parameters and thereafter emits a follow-up response in the presence of any ambiguities. Two example scenarios are illustrated in Figure 10. Sample scenario configuration along with extracted task parameters have been arranged in Table 5. The *Structured Information* column exhibits the information extracted from the Pointing State and the Verbal Command. The first column indicates whether the user was pointing or not, the second column enumerates experiments (shortened to "Exp") for corresponding pointing states, the third column lists different verbal commands with fixed task action *"bring"*, the fourth presents the extracted information from the verbal commands and the simultaneous pointing state, the fifth column arranges the predicted object of interest (OOI) that needs the action to be performed upon, and the sixth column tabulates the corresponding response formulated by the system. Light blue cells indicate the presence of ambiguity.

Table 2: Extracted task parameters from different verbal commands

Verbal command: "give me the plate"
Object: plate | Action: give | Attributes: None | Position: None

Verbal command: "bring me that red cup"
Object: cup | Action: bring | Attributes: [red] | Position: None

Verbal command: "go left"
Object: None | Action: go | Attributes: None | Position: left

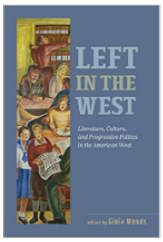
Verbal command: "grab the large green box on your right"
Object: box | Action: grab | Attributes: [green, large] | Position: right

Verbal command: "put the jar on the table"
Object: jar | Action: put | Attributes: None | Position: None

Table 3: Stored sequential task parameters

NO	Object	Action	Attributes	Position
1	plate	give	None	None
2	cup	bring	[red]	None
3	None	go	None	left
4	box	grab	[green, large]	right
5	jar	put	None	None

Object: cheez-it
Attribute: red



Object: book-1
Attribute: blue

Object: book-2
Attribute: red

Figure 9: Object attributes

Table 4: Pointing gesture recognition

Distance	Accuracy	Precision	Recall
4.88	1	1	1
3.66	0.995	1	0.99
2.44	0.995	1	0.99
1.22	0.995	1	0.99

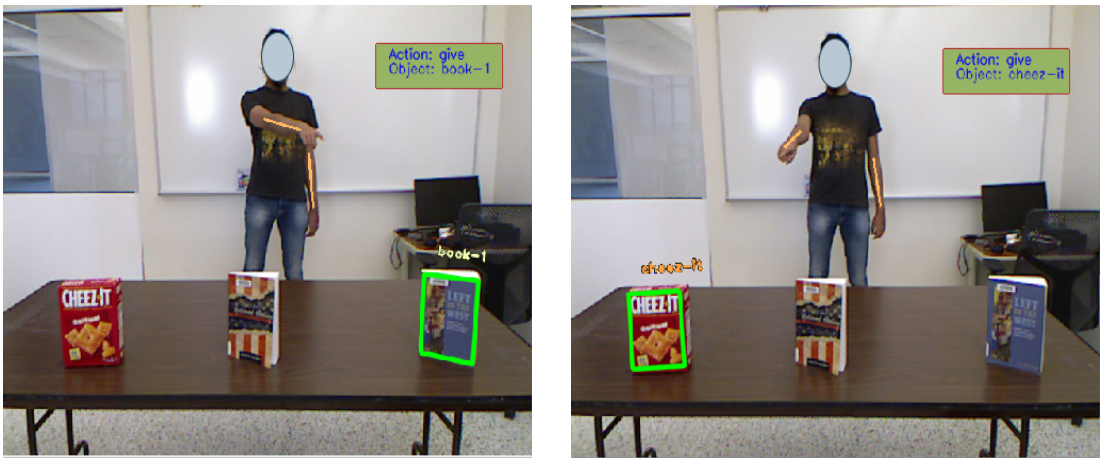
Ambiguity occurs when the **OOI** cannot be determined from the given verbal command and pointing gesture; the system fails to identify the object of interest and responds with the feedback *"Need additional information to identify object"*. Subsequently, the system waits for the user to perform the pointing gesture and/or amend the command; once these inputs are received, it repeats the entire process.

From the Table 5 we can see, for the *"Not Pointing"* state, ambiguity occurred when there was a lack of object attribute(s) (Exp 1, 3) for uniquely identifying the **OOI**, and thus, the system asked for additional information. For the *"Pointing"* state, the ambiguity arises when the pointing direction does not intersect with any of the object boundaries; in these scenarios, verbal commands can alleviate the ambiguity by providing additional information. Ambiguity can also arise if the inferred objects from the extracted identifiers and the pointing gesture are different. However, the system prioritizes the object inferred from the pointing gesture as the speech-to-text module may sometimes miss transcribe.

5 Conclusion and Further Research

In this paper, we have presented a HRI framework for extracting human-robot collaborative task parameters taking multiple inputs, and processing them simultaneously in real-time. Verbal communication is used to extract intricate information about the task e.g. action command, object attributes, etc., which is supported by pointing gesture recognition, the general direction along pointed object estimation to facilitate a natural interaction interface for the user. This information is assembled into a set of named parameters and can then be further analyzed to form a structured command that can be passed to and easily translated by a robotic entity.

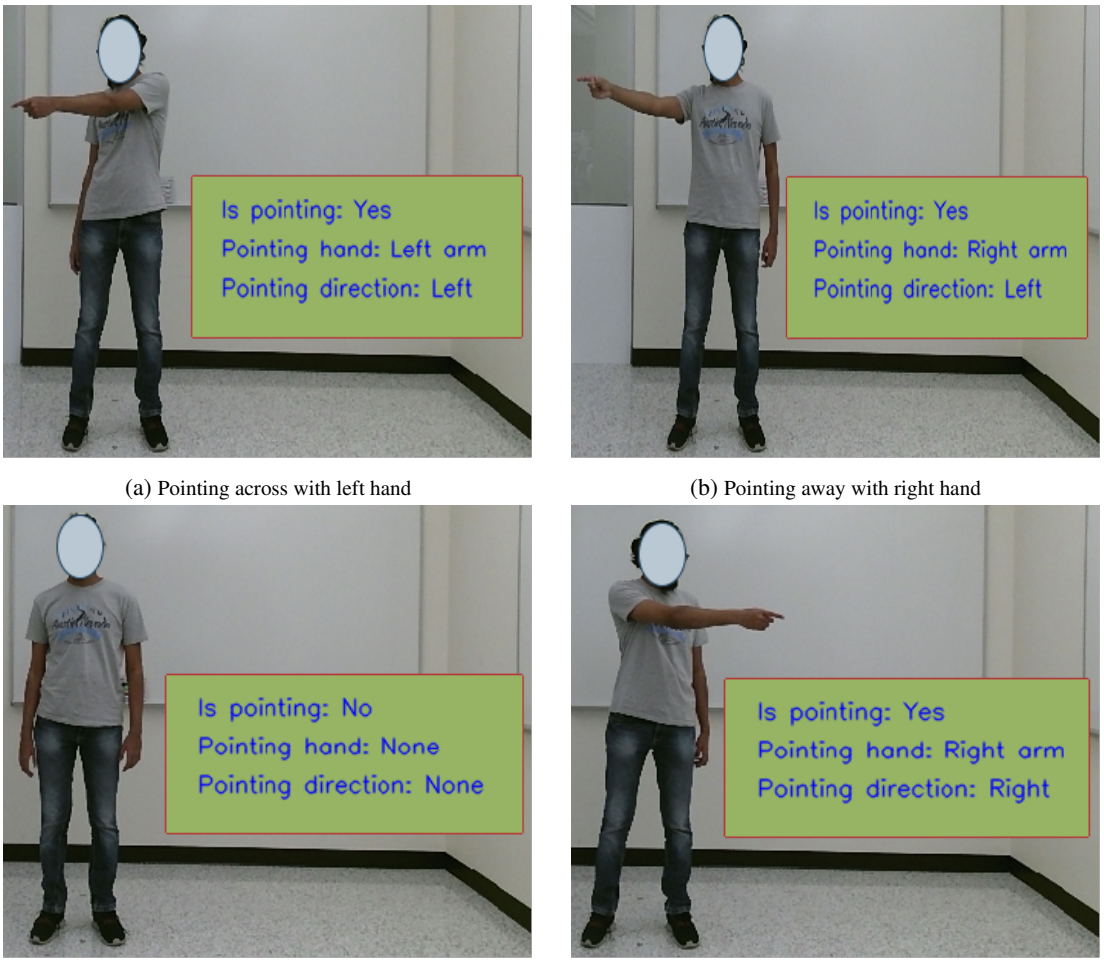
We presented a pointing gesture recognition approach from a 2D image frame that can detect the gesture, and estimate the pointing direction and the pointed object in the scene. We implemented a template matching based object detection and planar pose estimation technique that provides information about the object present in the scene. We have also devised two approaches to extract the task information prevalent in collaborative interactions. The verbal command is received from the sensor and then converted to text to further match with the previously formulated language patterns to extract relevant task specific parameters. All this information is integrated to form the final task configuration; if an essential parameter is missing or there are ambiguities, the system responds with appropriate feedback.



(a) Pointing to the object labeled "**book-1**"

(b) Pointing to the object labeled "**cheez-it**"

Figure 10: Example scenarios where the user points to different objects while voicing the command *"give me that"*



(a) Pointing across with left hand

(b) Pointing away with right hand

(c) No pointing

(d) Pointing across with right hand

Figure 11: System output with different pointing scenarios

Table 5: Generated task parameters

Pointing State	Exp #	Verbal Command	Structured information	Identified Object	Feedback
Pointing	1	bring that, bring me that	*{action: "bring", pointing_identifier: True, object: "book", object_identifiers: {attributes: null, position: null}}*	"book-1"	None
	2	bring the red book	*{action: "bring", pointing_identifier: True, object: "book", object_identifiers: {attributes: "red", position: }}*	"book-2"	None
	3	bring that red thing	*{action: "bring", pointing_identifier: True, object: null, object_identifiers: {attributes: "red", position: }}*	"cheez-it"	None
Not Pointing	1	bring that, bring me that	*{action: "bring", pointing_identifier: False, object: null, object_identifiers: {attributes: null, position: null}}*	None (ambiguous)	"Need additional information to identify object"
	2	bring the red book	*{action: "bring", pointing_identifier: False, object: "book", object_identifiers: {attributes: "red", position: null}}*	"book-2"	None
	3	bring that red thing	*{action: "bring", pointing_identifier: False, object: null, object_identifiers: {attributes: "red", position: "right"}}*	None (ambiguous)	"Need additional information to identify object"

We carried out experiments to measure the performance of our pointing gesture recognition system at different distances and it was able to recognize the state of the pointing gesture with very high accuracy. Next, we investigated the formation of the task configurations by passing different natural language instructions along with different gesture states. We have tabulated the extracted task parameters for different verbal commands along with how the task instructions are stored in a sequential list. Subsequently, we have illustrated the integration of these sensor data and interaction information with sample experimental results. Finally, we have listed the final task configurations along with the system feedback for different interaction scenarios.

The system can be further improved by introducing reliable 3D information. Having reliable depth information will effectively eliminate these limitations. Furthermore, more complex interaction scenarios comprising of multiple users and more intricate dialogues can be investigated to further develop more meaningful HRI systems.

References

[1] R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz, “"Tell Me When and Why to Do It! Run-Time Planner Model Updates via Natural Language Instruction",” in *"Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction"*, 2012, pp. 471–478.

[2] L. Dipietro, A. M. Sabatini, and P. Dario, “"A Survey of Glove-Based Systems and Their Applications",” *"Ieee Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)"*, vol. 38, no. 4, pp. 461–482, 2008.

[3] D. Droeschel, J. St"uckler, and S. Behnke, “"Learning to Interpret Pointing Gestures With a Time-of-Flight Camera",” in *"Proceedings of the 6th International Conference on Human-Robot Interaction"*, 2011, pp. 481–488.

[4] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn, “"What to Do and How to Do It: Translating Natural Language Directives Into Temporal and Dynamic Logic Representation for Goal Management and Action Execution",” in *"2009 IEEE International Conference on Robotics and Automation"*. IEEE, 2009, pp. 4163–4168.

[5] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, “"",” in *"Iccv"*, 2017.

[6] M. A. Fischler and R. C. Bolles, “"Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography",” *"Commun. ACM"*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: https://doi.org/10.1145/358669.358692

[7] S. Hochreiter and J. Schmidhuber, “"Long Short-Term Memory",” *"Neural Computation"*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] S. Iba, J. M. V. Weghe, C. J. Paredis, and P. K. Khosla, “"An Architecture for Gesture-Based Control of Mobile Robots",” in *"Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots With High Intelligence and Emotional Quotients (Cat. No. 99CH36289)"*, vol. 2. IEEE, 1999, pp. 851–857.

[9] N. Jojic, B. Brumitt, B. Meyers, S. Harris, and T. Huang, “"Detection and Estimation of Pointing Gestures in Dense Disparity Maps",” in *"Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)"*. IEEE, 2000, pp. 468–475.

[10] R. E. Kahn and M. J. Swain, “"Understanding People Pointing: The Perseus System",” in *"Proceedings of International Symposium on Computer Vision-Iscv"*. IEEE, 1995, pp. 569–574.

[11] R. E. Kahn, M. J. Swain, P. N. Prokopowicz, and R. J. Firby, “"Gesture Recognition Using the Perseus Architecture",” in *"Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition"*. IEEE, 1996, pp. 734–741.

[12] R. Kehl and L. Van Gool, “"Real-Time Pointing Gesture Recognition for an Immersive Environment",” in *"Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings."*. IEEE, 2004, pp. 577–582.

[13] T. Kollar, S. Tellex, D. Roy, and N. Roy, “"Toward Understanding Natural Language Directions",” in *"2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)"*. IEEE, 2010, pp. 259–266.

[14] Y.-L. Kuo, B. Katz, and A. Barbu, “"Deep Compositional Robotic Planners That Follow Natural Language Commands",” in *"2020 IEEE International Conference on Robotics and Automation (ICRA)"*. IEEE, 2020, pp. 4906–4912.

[15] J. Li, C. Wang, H. Zhu, Y. Mao, H.-S. Fang, and C. Lu, “"CrowdPose: Efficient Crowded Scenes Pose Estimation and a New Benchmark",” *"arXiv Preprint arXiv:1812.00324"*, 2018.

[16] D. G. Lowe, “"”,” *"International Journal of Computer Vision"*, 2004.

[17] M. MacMahon, B. Stankiewicz, and B. Kuipers, “"Walk the Talk: Connecting Language, Knowledge, and Action in Route Instructions",” *"Def"*, vol. 2, no. 6, p. 4, 2006.

[18] C. Matuszek, D. Fox, and K. Koscher, “"Following Directions Using Statistical Machine Translation",” in *"2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)"*. IEEE, 2010, pp. 251–258.

[19] M. Muja and D. G. Lowe, “"Fast Approximate Nearest Neighbors With Automatic Algorithm Configuration",” in *"International Conference on Computer Vision Theory and Application VISSAPP('09)"*. INSTICC Press, 2009, pp. 331–340.

[20] K. Nickel and R. Stiefelhagen, “"Pointing Gesture Recognition Based on 3d-Tracking of Face, Hands and Head Orientation",” in *"Proceedings of the 5th International Conference on Multimodal Interfaces"*, 2003, pp. 140–146.

[21] C.-B. Park and S.-W. Lee, “"Real-Time 3D Pointing Gesture Recognition for Mobile Robots With Cascade HMM and Particle Filter",” *"Image and Vision Computing"*, vol. 29, no. 1, pp. 51–63, 2011.

[22] M. Pateraki, H. Baltzakis, and P. Trahanias, “"Visual Estimation of Pointed Targets for Robot Guidance via Fusion of Face Pose and Hand Orientation",” *"Computer Vision and Image Understanding"*, vol. 120, pp. 1–13, 2014.

[23] S. K. Paul, M. T. Chowdhury, M. Nicolescu, M. Nicolescu, and D. Feil-Seifer, “"Object Detection and Pose Estimation From RGB and Depth Data for Real-Time, Adaptive Robotic Grasping",” in *"Advances in Computer Vision and Computational Biology"*. Springer, 2021, pp. 121–142.

[24] F. H. Previc, “"The Neuropsychology of 3-D Space.",” *"Psychological Bulletin"*, vol. 124, no. 2, p. 123, 1998.

[25] D. L. Quam, “"Gesture Recognition With a Dataglove",” in *"IEEE Conference on Aerospace and Electronics"*. IEEE, 1990, pp. 755–760.

[26] S. S. Rautaray and A. Agrawal, “"Vision Based Hand Gesture Recognition for Human Computer Interaction: A Survey",” *"Artificial Intelligence Review"*, vol. 43, no. 1, pp. 1–54, 2015.

[27] J. Richarz, A. Scheidig, C. Martin, S. M"uller, and H.-M. Gross, “"A Monocular Pointing Pose Estimator for Gestural Instruction of a Mobile Robot",” *"International Journal of Advanced Robotic Systems"*, vol. 4, no. 1, p. 17, 2007.

[28] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock, “"Spatial Language for Human-Robot Dialogs",” *"IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)"*, vol. 34, no. 2, pp. 154–167, 2004.

[29] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy, “"Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation",” in *"Proceedings of the AAAI Conference on Artificial Intelligence"*, vol. 25, no. 1, 2011.

[30] H. Watanabe, H. Hongo, M. Yasumoto, and K. Yamamoto, “"Detection and Estimation of Omni-Directional Pointing Gestures Using Multiple Cameras.",” in *"Mva"*, 2000, pp. 345–348.

[31] A. D. Wilson and A. F. Bobick, “"Parametric Hidden Markov Models for Gesture Recognition",” *"IEEE Transactions on Pattern Analysis and Machine Intelligence"*, vol. 21, no. 9, pp. 884–900, 1999.

[32] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, “"Pfinder: Real-Time Tracking of the Human Body",” *"IEEE Transactions on Pattern Analysis and Machine Intelligence"*, vol. 19, no. 7, pp. 780–785, 1997.

Shuvo Kumar Paul received his M.S. in 2020 in computer science and engineering from University of Nevada, Reno. He completed his B.S. from North South University, Bangladesh. Before joining UNR, he worked as a research associate at the AGENCY lab (previously CVCR). His research interests include machine learning, computer vision, computational linguistics, and robotics

Pourya Hoseini received his Ph.D. in 2020 and M.S. in 2017, both in computer science and engineering from University of Nevada, Reno. He also received a M.S. degree in 2011 and a B.S. in 2008 in electrical engineering from Urmia University and Azad University, Iran, respectively. His research interests are machine learning, computer vision, and evolutionary computing.

Arjun Vettath Gopinath is currently working as a Software Developer at N2N Services inc. He graduated with a degree in M.S. in Computer Science and Engineering from the University of Nevada, Reno in May 2021. He received his B.S. from SCMS School of Technology and Management, Kerala, India. His interests are Machine Learning, Human-Computer Interaction and Software Development.

Mircea Nicolescu is a Professor of Computer Science and Engineering at the University of Nevada, Reno and co-director of the UNR Computer Vision Laboratory. He received a PhD degree from the University of Southern California in 2003, a MS degree from USC in 1999, and a BS degree from the Polytechnic University Bucharest, Romania in 1995, all in Computer Science. His research interests include visual motion analysis, perceptual organization, vision-based surveillance, and activity recognition. Dr. Nicolescu's research has been funded by the Department of Homeland Security, the Office of Naval Research, the National Science Foundation and NASA. He is a member of the IEEE Computer Society.

Monica Nicolescu is a Professor with the Computer Science and Engineering Department at the University of Nevada, Reno and is the Director of the UNR Robotics Research Lab. Dr. Nicolescu earned her PhD degree in Computer Science from the University of Southern California (2003) at the Center for Robotics and Embedded Systems. She obtained her MS degree in Computer Science from USC (1999) and a BS in Computer Science at the Polytechnic University Bucharest (Romania, 1995). Her research interests are in the areas of human-robot interaction, robot control, learning, and multi-robot systems. Dr. Nicolescu's research has been supported by the National Science Foundation, the Office of Naval Research, the Army Research Laboratory, the Department of Energy and Nevada Nanotech Systems. In 2006 she was a recipient of the NSF Early Career Development Award (CAREER) Award for her work on robot learning by demonstration

InFra_OE: An Integrated Framework for Ontology Evaluation

Narayan C. Debnath*, Archana Patel*, Debarshi Mazumder*, Phuc Nguyen Manh*, Ngoc Ha Minh*
Eastern International University, Binh Duong, VIETNAM

Abstract

Nowadays, ontologies are used everywhere to share information semantically, and hence it is crucial to evaluate before using them. Ontology evaluation becomes more important when we have more than one ontology for a domain and we have to choose one ontology from them. The existing ontology evaluation approaches focus on only a few ontology evaluation criteria. Therefore, they cannot determine the overall quality of the ontology. This paper aims to propose an integrated framework for the evaluation of ontologies. The proposed framework uses a knowledge representation approach, criteria-based approach, software engineering approach, and layer-based approach to evaluate the quality of the ontologies based on various criteria.

Keywords: Ontology, ontology evaluation, semantic, knowledge representation, OOPS!

1 Introduction

Knowledge representation and reasoning is a field of 'Artificial Intelligence' that encodes knowledge, beliefs, actions, feelings, goals, desires, preferences, and all other mental states in the machine. Nowadays, ontology is prominently used to represent knowledge. Earlier than ontologies, semantic network and semantic frame emerged, but they were lacking in formal semantics despite the fact that they had semantic in the name [30]. Ontologies offer the richest machine-interpretable (rather than just machine-processable) and explicit semantics and are being used today extensively for semantic interoperability and integration. Ontology is a knowledge representation formalism that reduces the problem of big semantic loss in the process of modelling knowledge [24]. Ontology does not only provide sharable and reusable knowledge, but it also provides a common understanding of the knowledge; as a result, the interoperability and interconnectedness of the model make it priceless for addressing the issues of querying data. Ontology work with concepts and relations that are very close to the working of the human brain. It also provides a way to represent any data format like unstructured, semi-structured, structured, and enables data fetching with semantics. The key requirement of ontology is the development of suitable languages for the representation and extraction of information. Varieties of ontology languages have been developed, and the most operable and standard language is web ontology language (OWL) [18]. Ontology query language plays a very important role in extracting and processing the information. SPARQL is one of the most widely used ontology query languages [25]. By using these semantic technologies (Ontology, SPARQL, OWL), users and systems can interact and share information with each other in an intelligent manner.

Ontology can be developed either from scratch or by modifying an existing ontology [28]. Many well-known ontology repositories are available that contain more than a thousand ontologies about a domain, such as-

- OBO Foundry: It contains biological science-related ontologies
- Bio portal: It is a comprehensive repository of biomedical ontologies
- Agro portal: It is a vocabulary and ontology repository for agronomy related domains
- OLS: It provides single-point access to the latest version of biomedical ontologies

Users use these repositories to choose the ontology for an application. Ontology evaluation is a way that determines the relevance and importance of the ontology in a specified domain [35]. Ontology evaluation is an essential process for the development and maintenance of an ontology. Mainly, we need to evaluate the ontology because of two reasons:

1. The developed ontology needs to be evaluated to check the quality of the ontology. The ontology evaluation is also a phase of the ontology development life cycle.
2. For the reusability purpose, we work with the existing ontologies. However, when more than one ontology is available for a domain, then it is hard to choose one ontology among them. In such a case, we need to evaluate the ontology to find the best-suited ontology according to the need.

The work of ontology evaluation is focused on five questions [28]. What should be evaluated? Why should it be evaluated? When should it be evaluated? What should be the base of the evaluation? What are the possible criteria to evaluate the design and implementation of an ontology? The various tools

* Department of Software Engineering, Email: narayan.debnath@eiu.edu.vn, archana.patel@eiu.edu.vn, debarshi.mazumder@eiu.edu.vn, phuc.nguyenmanh@eiu.edu.vn, ngoc.ha@eiu.edu.vn.

are proposed in the literature to evaluate the ontology; some are available on the web. These tools are grouped into two categories: domain-dependent ontology evaluation tools and domain-independent ontology evaluation tools. These tools evaluate the design of ontologies based on various criteria like, accuracy, adaptability, cognitive adequacy, completeness, clarity, expressiveness, conciseness, consistency, and grounding. In contrast, computational efficiency, precision, recall, and practical usefulness are used to evaluate the implementation of an ontology. Many studies are available for the evaluation of the ontology; however, they have some limitations:

1. The existing studies usually show only a set of criteria and questions, and they do not show the guidelines to evaluate the ontology.
2. The effort to evaluate the ontology is very high as no tool available that concisely evaluates the ontology.
3. The evaluation heavily depends on the evaluator's expertise to understand the evaluation criteria and questions.
4. The evaluation is still very subjective.

The proposed work presents an integrated framework for ontology evaluation called InFra_OE. The framework takes into account four fundamental principles: (a) It supports five roles of knowledge representation proposed by Davis [10] (b) It is based on the evaluation of the effectiveness of the coding standard (a software engineering approach) (c) It integrates the OOPS! tool for anomalies detection (d) It uses a layer-based approach. The main contributions of this work are: (a) to propose an integrated framework for ontology evaluation that integrate the features of four approaches, namely five roles of knowledge representation, approach for evaluation of software development methodology, ontology pitfall scanner (OOPS!) tool, and different layers of ontologies. (b) to propose an algorithm for the step-by-step execution of the proposed framework. The rest of the paper is organized as follows: Section 2 shows the definitions of important concepts that are required to understand the proposed framework. Section 3 discusses the available literature of ontology evaluation. Section 4 describes the proposed framework and algorithm. The last section concludes the paper.

2 Background

Ontology Scope: The first step of ontology development is to determine the scope of an ontology. In ontological engineering, ontology scope shows the specification and design aspects for the representation of the knowledge [12]. The scope of an ontology can be classified into three aspects: Domain scope (determines that the scope of the ontology is relevant to the task for which ontology is designed), conceptual scope (determines that ontology will represent the hierarchical and taxonomical concepts), technical scope (shows that the specifications and requirements for ontologies are integrated smoothly and correctly in terms of ontology integration and application in practice).

Ontology Layers: The structure of the ontology is complex, and it is hard to evaluate the whole ontology once. So, it is required to evaluate the ontology according to its different layers. The layer-based ontology evaluation approach allows users to use different techniques for different layers [19]. Mainly, ontology has four different layers: Lexicon/vocabulary layer (This layer evaluates the ontology with respect to knowledge representation and conceptualization of ontologies like naming criteria for concepts, instances, and facts.), structure/ architecture layer (this layer evaluates the hierarchical and taxonomic elements of ontology like hierarchical relations among concepts). Representation/semantic layer (this layer evaluates the ontology with respect to the semantic elements), context/application layer (This layer evaluates the ontology according to the context and application where the ontology would be used. Typically, evaluation looks at how the outcomes of the application are affected by the use of ontology).

Type of Ontology: Mainly, four types of ontologies are available, namely upper ontology, domain ontology, task ontology, and application ontology. The *upper ontology* occupies general concepts or terms like matter, time, and space. The aim of the upper ontologies is to support broad semantic interoperability among domain ontologies by providing a common platform for the formulation of the definitions. The *domain ontology* is developed to capture and relate the content of specific domains (e.g., medical, electronic, digital domain) or part of the world. Domain ontologies use the services of upper ontologies. The *task ontology* contains fundamental concepts according to a general activity or task. It is a specification of element relationships of tasks to explain how tasks can exist and be used in a specific environment. Task ontology serves as a foundation for using tasks in certain fields, like in the field of management, and it defines what element it has and what type of relationships can be established with other tasks. The *application ontology* is a specialized ontology focused on a specific application. It has a very narrow context and limited reusability because it depends on the particular scope and requirements of a specific application. Application ontologies are typically developed ad hoc by the application designers.

Ontology Development Methodology: In the literature, various authors have developed ontologies for the semantical analysis of data [9]. However, the major problem for ontology developers is to choose the right methodology that builds correct, complete, and concise ontology as per requirement. Ontology development methodology describes the step-by-step process for ontology development. The most famous used methodologies are TOVE, Enterprise Model Approach, METHONTOLOGY, and KBSI IDEF5. These methodologies have various steps for ontology development, and some steps are common among them. Figure 1 shows the relationship among these methodologies via arrows (double-headed arrows) [11].

The most important step of ontology development is to identify the purpose and fix the boundary/scope of the ontology. This can be achieved by writing competency questions and the ontology and impose constraints on the classes and their

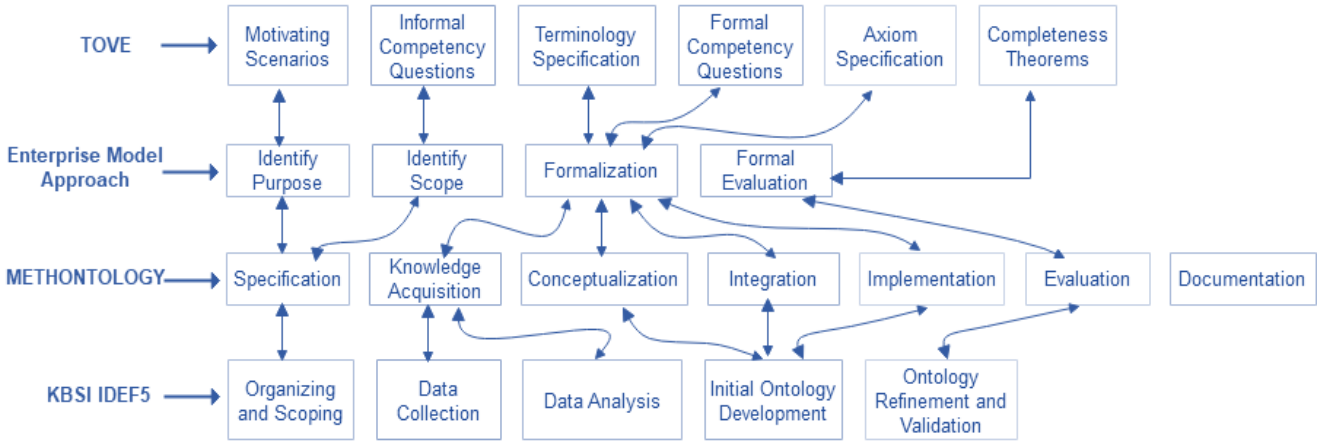


Figure 1: Most popular and extensively used ontology development methodologies

properties as required. Ontology evaluation is the vital step of ontology development methodologies. It shows the quality of the developed ontology based on various crietria like completeness (ontology must contain all the required information as per domain need), and accuracy (ontology must be free from anomalies and is able to infer the correct answer). The last step of ontology development methodology is to document the ontologies that can become the base of other activities. Apart from these methodologies, three ontology development methodologies, namely Neon, YAMO, and SAMOD are also available in the literature [11].

Ontology Evaluation Quality Criteria: Ontology evaluation checks the quality and quantity of an ontology based on various criteria [23]. The essential criteria are:

- Accuracy: It is determined by the definitions, descriptions of entities like classes, properties, and individuals. This criterion states that the ontology is correct.
- Clarity: Clarity evaluates how well an ontology communicates the intended meaning of specified concepts. Clarity is determined by a number of factors. First and foremost, definitions must be objective and independent of social or computational circumstances. Social events or computational needs may motivate the definition of a notion. Second, ontologies should utilize definitions for classes rather than descriptions. Third, entities should be adequately documented and completely labeled in all essential languages, among other things. Most of these criteria are best examined using criterion-based techniques such as OntoClean.
- Completeness: It states that the ontology covers complete information about a specified domain. Sometimes some important information about the entities is missing in the ontology, which leads to an ambiguity problem and hampers the results of reasoning. Precision and recall are measured to check the incompleteness problem in the ontology. Precision shows the ability of an ontology to present only relevant items, whereas recall shows the ability of an ontology to present all relevant items. The completeness of entities depends on the level of granularity agreed to in the whole ontology.
- Adaptability: It measures the adaptability of an ontology and shows how far the ontology anticipates its uses. An ontology should offer the conceptual foundation for a range of anticipated tasks.
- Consistency/Coherence: Consistency highlights the fact that the ontology does not include or allow for any inconsistencies. An ontology should be coherent, which means that it should allow inferences that are compatible with the definitions. The defining axioms should be logically consistent. Coherence should also apply to imprecisely defined ideas, such as those given in natural language documentation. An example of a contradiction is the element Lion's description, “A lion is a giant cat that lives in pride,” while possessing a logical axiom ClassAssertion(ex: Type of chocolate ex: Lion). Consistency can be evaluated using criteria-based techniques that focus on axioms. It can also be recognized depending on the ontology's performance in a specific job.
- Reusability: The feature indicates that good ontologies immediately lead to increased data reuse and improved collaboration across application and domain boundaries. The reusability of an ontology may be determined by analysing the various metadata that is available for it or by investigating the interactions within its specific community.
- Computational efficiency: It shows the flexibility of an ontology with the tools, specifically focusing on the speed of the reasoner that infers the information from the ontology.
- Extendibility: Extendibility defines the high-level needs of an ontology design that must be specialized enough for usability, extendable for upgrades, abstract enough for

reusability, and compatible with current applications even after future modifications. An ontology is not anticipated to contain all of the potential information, attributes, and constraints of the domain it represents. As a result, several versions addressing the same body of information are feasible as far as the ontology design meets specified standards. In general, a good ontology holds adequate knowledge to develop expertise in solving significant issues.

- Minimal encoding bias: This criterion represents that encoding bias should be minimized because knowledge-sharing agents may be developed with a variety of libraries and representation styles. An encoding bias occurs when representation choices are selected purely for the sake of notation or implementation. This quality highlights the platform-independent representation of knowledge.
- Conciseness: It examines the usefulness and preciseness of the stored information in an ontology. An ontology is called precise if it does not store any useless or unnecessary definitions; if any explicit redundancies of definitions of entities and between definitions of entities do not exist.
- Minimal ontological commitment: This criterion evaluates that the ontology should define just those terms that are necessary for communicating information compatible with that theory. Nonrelevant information should not be part of the top-level ontology. Top-level ontology can be further specialized by the respective community.
- Expandability: This shows the effort that needs to be put to add new definitions and more knowledge to an entity of an ontology without altering the set of well-defined properties already guaranteed.
- Sensitiveness: This examines how small changes in a definition of an entity alter the set of well-defined properties already guaranteed.
- Organizational fitness: It investigates how ontology is easily deployed within the organization.
- Agreement: Measured through the proportion of agreement that experts have with respect to ontology elements, that is, by measuring the consensus of a group of experts.

Table 1: Most commonly identified ontology evaluation criteria

	Thomas Gruber	**Gómez Pérez**	**Denny Vrandecic**	**Gangemi**
Clarity	√		√	
Consistency		√	√	
Coherence	√		√	
Extendibility	√			
Completeness		√	√	
Minimal encoding bias	√			
Conciseness		√	√	
Minimal ontological commitment	√			
Expandability		√		
Sensitiveness		√		
Accuracy				
Adaptability			√	
Computational efficiency			√	
Organizational fitness			√	
Agreement				√
User Satisfaction				√
Task				√
Topic				√
Modularity				√

- User Satisfaction: This can be evaluated by dedicated research or reliability assessment
- Task: This deals with measuring an ontology according to its fitness to some goals, postconditions, preconditions, options, constraints, and others.
- Topic: This measures the ontology according to its fitness for a repository of existing knowledge.
- Modularity: Modularity measures fitness to a repository of existing reusable components.

3 Related Work

There are a lot of studies available for ontology evaluation. These studies are grouped into two categories: domain-dependent ontology evaluation studies and domain-independent ontology evaluation studies. The domain-dependent ontology evaluation studies show the evaluation of domain dependant ontologies and discuss the available tool for the same. The domain-independent ontology evaluation studies show the evaluation of domain-independent ontologies and discuss the available tool for the same [25]. Figure 2 shows the domain dependent and independent tools.

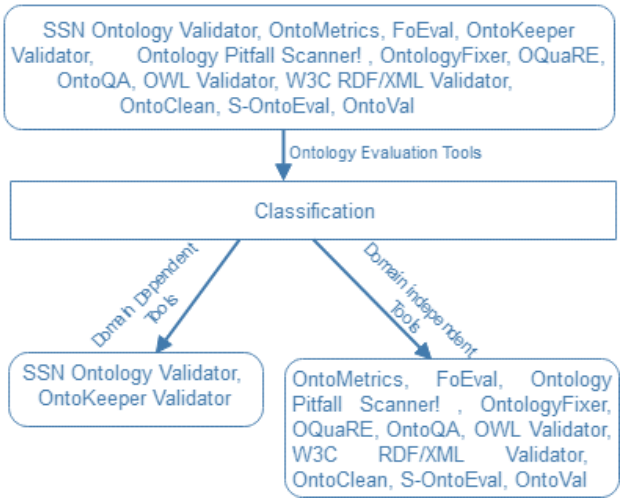


Figure 2: Ontology evaluation tools

SSN Ontology Validator [21] is used to validate the new ontology of the IoT domain. The validator gives a validation report that reports inconsistencies in an ontology. The SSN validator receives the ontology and compares it with the SSN ontology and other ontologies associated with it. *OntoKeeper Validator* [1] evaluates biodiversity ontologies with different levels of granularity. It analyzed the ontology files using semiotic metrics (semiotics has three branches, namely, syntactic, pragmatic, and semantic) and detects the quality score of the biodiversity ontology based on various parameters, namely richness, interpretability, comprehensiveness, and accuracy. *OWL Validator* [17] aims to ensure that all concepts and properties in the ontology are specified as per the W3C standard. OWL validator shows an error message with the detailed report when an input ontology does not support the selected profile. *W3C RDF/XML validator* [33] validates the RDF document by tracking the RDF issues and shows a warning message when an error occurs. W3C RDF validator shows the number of tuples (subject-object-predicate) that are encoded in the ontology as well as its graphical representation. This validator aims to ensure that the document is syntactically valid.

Ontology Pitfall Scanner! (OOPS!) [34] tool shows the pitfalls or anomalies of an ontology. OOPS! shows the 41 types of pitfalls and groups them into three categories, namely, minor pitfalls (these pitfalls are not serious and no need to remove), important pitfalls (not very serious pitfalls but need to remove), critical pitfalls (these pitfalls hamper the quality of an ontology and need to remove them before using the ontology). *OntoMetrics tool* [22] calculates the statistical information about an ontology. It has five types of metrics, namely, Base metric, Schema metrics, Knowledge base metric, Class metric, and Graph metrics. *OntologyFixer tool* [31] allows the detection and correction of errors. It uses various metrics to measure the different aspects of the quality of an ontology. These metrics are ANOnto (shows annotation richness), CBOnto (shows the coupling between objects), CROnto (shows class richness), INROnto (shows the number of relations per class), LCOMOnto (shows lack of cohesion in methods), NOMOnto (shows the number of properties per class), RCOnto (shows the distribution of instance across class), RFCOnto (shows response measure for a class), and RROnto (shows relationship richness). To detect the pitfalls of an ontology, OntologyFixer integrated OOPS! tool. *OntoVal tool* [3] evaluates the OWL ontologies by nontechnical domain specialists and allows users to provide textual feedback for each evaluated term and evaluate the correctness of the developed ontology via an integrated engine. OntoVal starts the ontology evaluation by collecting information about the participant (name; age; domain experience level, ranging from 0 to 10; ontology experience level, ranging from 0 to 10). The ontology evaluation process of OntoVal is divided into three stages, namely, class evaluation, property evaluation, and overall evaluation. The *Semiotic-based Ontology Evaluation (S-OntoEval) tool* [13] aims to evaluate the quality of the ontology by taking three metrics, namely, syntactic, semantic, and pragmatic, that are considered different aspects of the ontology quality. In essence, there are three types of evaluation levels, namely, structural level, functional level, and usability related level.

OQuaRE framework [14] evaluates the quality of an ontology on the basis of both quality models and quality metrics. OQuaRE adapts and reuses five characteristics from the SQuaRE, namely, Structural (it specifies the formal and semantic important properties of an ontology), functional adequacy (it includes the degree of accomplishment of functional requirements), Reliability (it checks the level of performance under stated conditions), Operability (it shows the effort needed for building an ontology and individual assessment), Maintainability (it shows the ability of ontologies to be modified by changes in environments). The *Ontology Quality Analysis (OntoQA) approach* [32] evaluates the design

and representation of knowledge of ontology, and the placement of instances within the ontology and its effective usage. The OntoQA categorizes the quality of the ontology into three groups, namely, Schema metrics, Knowledgebase metrics, and Class metrics.

OntoClean Methodology [16] validates the adequacy of the ontology hierarchy based on general ontological notions, namely, essence, unity, and identity. These notions are used to characterize relevant aspects of the intended meaning of classes, properties, and relations. It checks the correctness of the ontology hierarchy via the principles of metaproperties, namely, rigidity (this property is essential to all their instances), unity (refers to being able to recognize all parts that form an individual entity), identity (refers that all instances are identified in the same way), and dependence (captures a meta-property of certain relational roles).

Full Ontology Evaluation (FoEval) model [6] is a ranking and selection tool that has three features, namely: it allows the user to select a set of metrics that help in the evaluation process, this tool enables the user to evaluate the locally stored and searched ontologies from different search engines or repositories, it captures the structural and semantic information of a domain. It includes a rich set of metrics, namely, coverage, richness, comprehensiveness, and computational efficiency.

4 Proposed Framework

The proposed framework consists of three phases, namely Input phase, Processing Phase, and Output phase, whereas the processing phase contains four modules (a) Evaluation based on Role of Knowledge Representation, (b) OOPS! (c) Ontology Code Effectiveness, and (d) Layer Based Evaluation. The human interaction is required in two modules, namely evaluation based on the role of knowledge representation and layer-based evaluation. All the modules of the processing phase are executed parallelly, and the final result is calculated by taking the mean value of obtained results from all four modules. Figure 3 shows the proposed integrated framework for ontology evaluation.

I. Input Phase: It consists of knowledge base and expert person. The knowledge base consists of domain ontologies, core ontologies, and upper ontologies. Any type of these ontologies can be input into the processing phase. The expert assigns the appropriate value of parameters of these ontologies as per need. The expert input is required in two modules of the processing phase. The proposed framework is called semiautomatic because of the involvement of the experts.

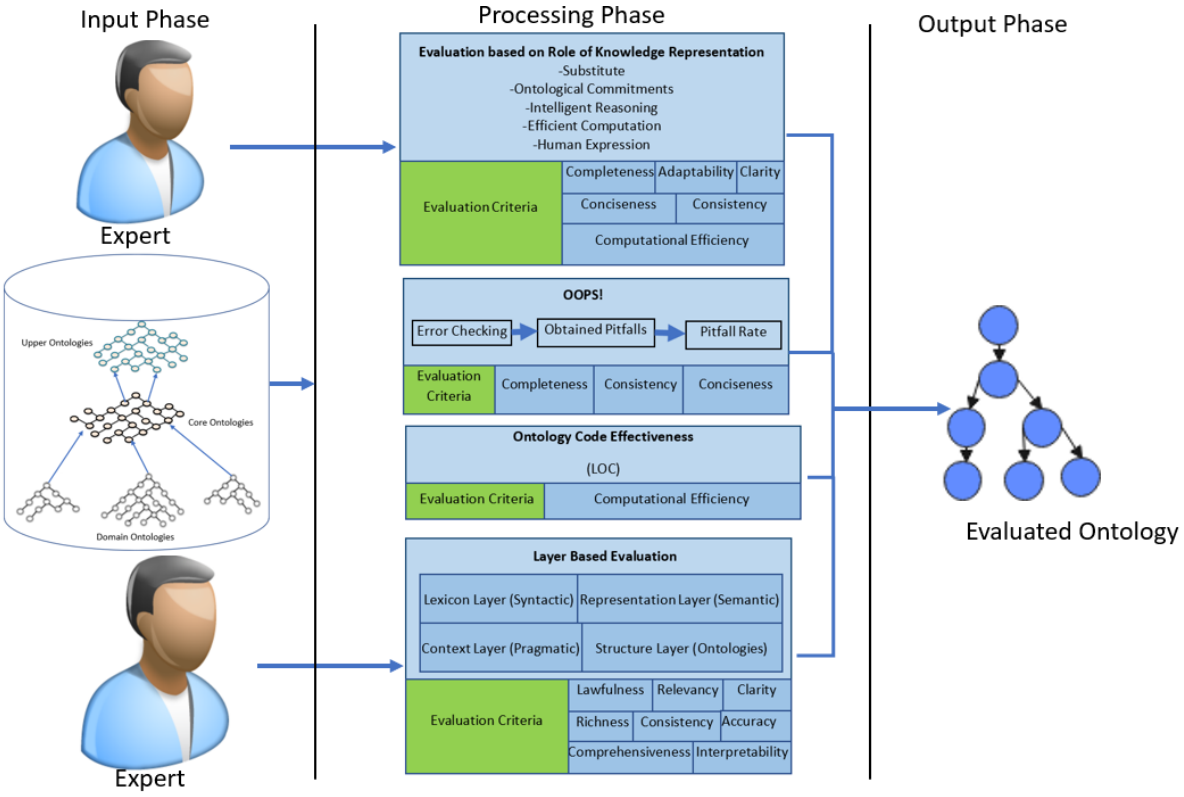


Figure 3: InFra_OE: An integrated framework for ontology evaluation

II. Processing Phase: This phase takes ontologies from the knowledge base and then executes all the four modules of the processing phase over it. These modules evaluate the ontologies from different aspects and generate different results for the ontology evaluation.

a) Evaluation based on Role of Knowledge Representation: Davis [10] has presented five roles for the discussion of knowledge representation. These roles clearly describe What is Knowledge Representation? We use these roles for the evaluation of ontology [4]:

- Substitute: This role shows how ontology approaches the real world. It focuses on which concepts should be represented and which should be omitted. For example, vehicle, rim, tires, and handlebars are the concepts of the bicycle. The concepts which are close to the real world need to be represented to fulfill this role of knowledge representation.
- Ontological Commitments: This role shows how the ontology is closer to the real world. This role will be fulfilled better if the representation is more consistent. For example, the more consistent representation is *a bicycle is a vehicle* and *a vehicle is an object* as compared to *bicycle is an object*.
- Intelligent Reasoning: This role represents how the ontology correctly infers the real world. This role will be fulfilled by the concise representation of the relations and attributes. For example, the vehicle is a bicycle because it has two tires, thin wheels, and handlebars.
- Efficient Computation: This role represents how the machine can think about a domain and extract the information within minimum time (i.e., computational time). Suppose, all the websites have a bicycle domain ontology, and the user is searching for a bicycle which has a red color, two tires of size x, and manufactured by company y, then the machine must be able to find this bicycle in a few seconds.
- Human Expression: This role represents how easy it is to understand the modelling. This role will be achieved by a clear declaration/representation of the concepts and relations. For example, the concept of bicycle is represented by bicycle, not any other words like bi or bic.

Each of these roles is fulfilled by some questions (shown in Table 2) and shows different criteria of ontology evaluation.

Table 2: Connection between role, questions, ontology development phases, and ontology evaluation criteria

Role	Questions	Ontology Development Phases	Ontology Evaluation Criteria
Substitute	Q1. Address the document that specify the scope and objective of the ontology.	Scope Determination, Concept Extraction, Encoding	Completeness, Adaptability
	Q2. Address the coherence between the document of Q1 and modelling of the ontology.		
	Q3. Address the reusability of the concepts that model the real world.		
Ontological Commitments	Q4. Address about the representation scheme for a specific domain	Concept Extraction and Encoding	Conciseness, Consistency
	Q5. Address about the representation scheme for an abstract domain		
	Q6. Address the coherence with the real world.		
Intelligent Reasoning	Q7. Address the reasoning power of ontology	Evaluation	Consistency
Efficient Computation	Q8. Address computational performance in term of successfully executed queries.	Evaluation	Computational efficiency
	Q9. Address computational performance in term of reasoner speed.		
Human Expression	Q10. Address the easy and precise understanding of the modelling.	Encoding	Clarity

The evaluation of an ontology with respect to five roles are calculated by the below mentioned equation [9].

$$\frac{\exp\{-0.44 + 0.03(Cov_S \times Sb)E + 0.02(Cov_C \times Co)E + 0.01(Cov_R \times Re)E + 0.02(Cov_{Cp} \times Cp)E - 0.66LExp_E - 25\ (0.1 \times Nl)E\}}{1 + \exp\{-0.44 + 0.03(Cov_S \times Sb)E + 0.02(Cov_C \times Co)E + 0.01(Cov_R \times Re)E + 0.02(Cov_{Cp} \times Cp)E - 0.66LExp_E - 25\ (0.1 \times Nl)E\}} \quad (1)$$

The parameter E indicates the evaluators/experts that assign the value of all the parameters and evaluates the quality of the ontology based on the five roles of knowledge representation.

- Cov_S shows the mean value of role 1 i.e. Substitute by rating all the three questions corresponding to role 1.
- Cov_C shows the mean value of role 2 i.e. Ontological Commitments by rating all the three questions corresponding to role 2.
- Cov_R shows the mean value of role 3 i.e. Intelligent Reasoning by rating the question corresponding to role 3.
- Cov_P shows the mean value of role 4 i.e. Efficient Computation by rating all the two questions corresponding to role 4.
- $LExp_i$ shows the value of the evaluator/expert experience, its value will be 1 if the expert has good knowledge about the ontologies otherwise 0.
- Nl value will be 1 if the expert will not be able to answer all the questions of the goal.
- The value of Sb = 1, Co = 1, Re = 1, Cp = 1, if total quality will be calculated otherwise it will be 0 according to the absence of any role which indicates the partial ontology evaluation.

(b) *OOPS!*: It is a web-based tool that shows the pitfalls or anomalies of an ontology. OOPS! shows the 41 types of different pitfalls starting from P01 to P41. Basically, OOPS! groups the pitfalls under three categories, namely minor pitfalls (these pitfalls are not serious and no need to remove), important pitfalls (not very serious pitfalls but need to remove), critical pitfalls (these pitfalls hamper the quality of an ontology and need to remove them before using ontology). The pitfall describes the number of features that could create problems during reasoning. We have calculated the pitfall rate by using the following equation [27]

$$A = \frac{\sum_{i=1}^{n} P_i}{N} \quad (2)$$

P_i represents the total number of pitfall cases according to the pitfall type P_i, and N is the total number of tuples (ontology size). The high value of the pitfall rate implies a more significant number of anomalies and vice-versa. To calculate the quality of the ontology, we subtract the obtained pitfall rate (A) by 1. It focuses on the completeness, consistency, and conciseness criteria of ontology evaluation.

(c) *Ontology Code Effectiveness*: Ontology code effectiveness evaluates the size and complexity of the ontology's code. Basically, it shows the conciseness of the developed ontology by identifying the similarity in an ontology code and identification of the duplicate in a code (known as clones). In software engineering, the quality of the code is determined by various techniques like Line of code (LOC), function point, etc. However, we evaluate the effectiveness of the ontology coding standard by the Goal-Question-Metric (GQM) approach proposed by Basili et al. [5]. Figure 4 shows the GQM approach. We calculate the ontology size by LOC, which shows the number of tuples stored in the ontology. The size of the ontology does not depend on the annotation properties as they serve metadata information about the entities. We have ignored nine annotation properties, namely- backwardCompatibleWith, comments, deprecated, incompatibleWith, isDefinedBy, label, priorVersion, seeAlso, and versionInfo. These properties are supported by the protégé tool.

(d) *Layer Based Evaluation*: The structure of the ontology is complex, and it is hard to evaluate the whole ontology at once. Hence, it is good practice to evaluate the ontology based on the layer approach. Mainly, ontology has four different layers [7]:

- Lexicon/Vocabulary layer: This layer evaluates the ontology with respect to knowledge representation and conceptualization of ontologies like naming criteria for concepts, instances, and facts. Ex- Bicycle or bic.
- Structure/ Architecture layer: This layer evaluates the hierarchical and taxonomic elements of ontology, like hierarchical relations among concepts. Ex- human must be a superclass of Male and Female.
- Representation/ Semantic layer: This layer evaluates the ontology with respect to the semantic elements. Ex- Mouse should be able to explain itself either mouse is a device or mouse is an animal.
- Context/Application layer: This layer evaluates the ontology according to the context and application where the ontology would be used. Typically, evaluation looks at how the outcomes of the application are affected by the use of ontology.

The advantage of the layer-based ontology evaluation approach is that it allows users to use different techniques at different ontology layers. We use the syntactic approach at Lexicon/Vocabulary layer; wordnet (lexical database of semantic relations between words) at the Structure/Architecture Layer; Semantic approach at the Representation/semantic layer; and Pragmatic approach at the Context/Application layer.

Syntactic Approach: It measures the quality of the ontology based on syntax and the way it is written. It focuses on syntactic correctness (checks how ontology language's rules are compiled), Richness (shows the number of ontological entities) [8]. The overall syntactic quality is calculated by below mentioned equation-

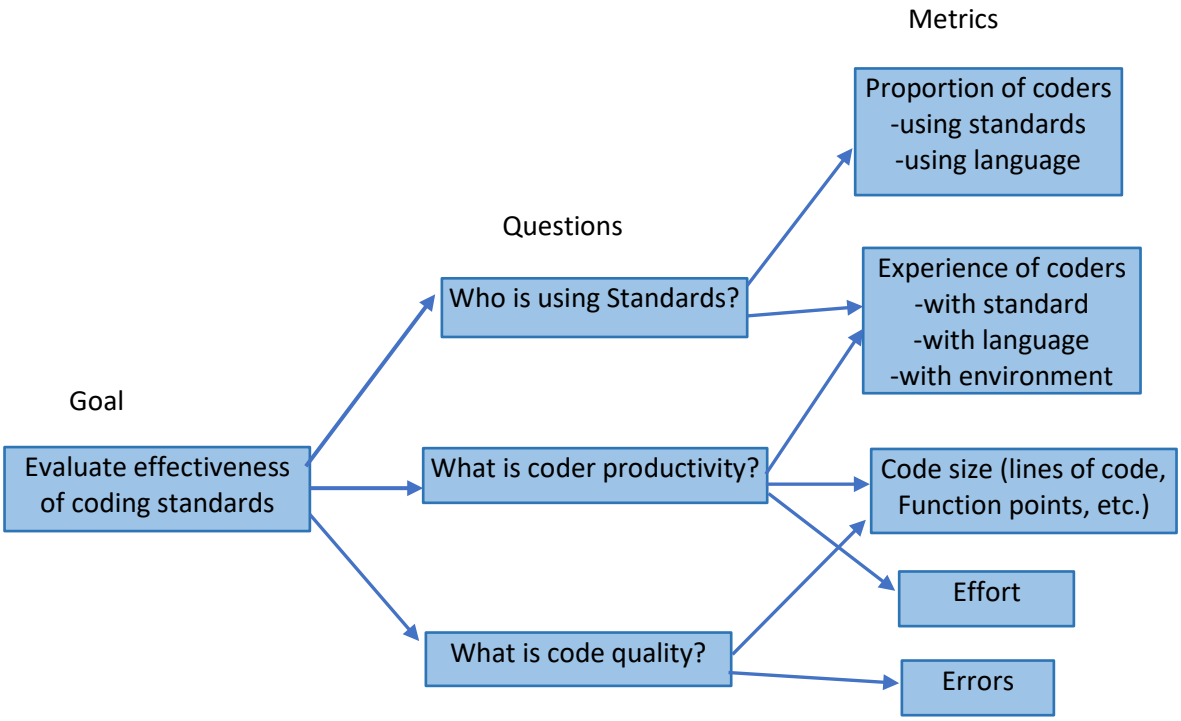


Figure 4: Goal-questions-metrics approach

$$Le = w_1 \times La + w_2 \times Ri \quad (3)$$

La= Number of violated axioms over the total number of axioms

Ri= Used number of ontological features over the total number of ontological features

- Semantic Approach: It deals with the meaning of the entity that is derived from the syntax via logic. It focuses on Interpretability (reveals that every ontological term shows correct meaning in everyday usage), Consistency (shows the ontological terms are uniformly defined and lack duplicate terms), Clarity (shows that all terms are unambiguously represented) [3]. The overall Semantic quality is calculated by below mentioned equation-

$$Se = w_3 \times In + w_4 \times CO + w_5 \times Cl \quad =(4)$$

In: Terms which has at least one-word sense over the total number of terms

CO: Number of duplicated terms divided by the total number of terms

Cl: the average word sense per term over the number of terms

- Pragmatic Approach: It deals with inferential meaning, not merely logical inference, but the subtler aspects of communication expressed through indirection [20]. It focuses on comprehensiveness (shows the coverage of domain), accuracy (shows the truthfulness of the statement), and relevancy (shows the relevancy of ontology in particular applications).

$$Con = w_6 \times Com + w_7 \times Ac + w_8 \times Re \quad (5)$$

Com= percentage number of instances, classes, and properties of the ontology to a group of ontologies.

AC= truth statement over the total statement.

Re= varies and depends on the possible use-case of the ontology.

III. Output Phase: This phase shows the numeric value for the evaluated results by taking the mean of all the four modules of process6ing, namely Evaluation based on Role of Knowledge Representation, OOPS!, Ontology Code Effectiveness, and Layer based approach. The obtained value lies between [0,1]. The highest value shows that ontology has good quality.

InFra_OE has mainly four functions KR(), OOPS(), OCE(), and LBE (). The function KR() corresponds to the five roles of knowledge representation (module 1 of phase 2) and returns an integer value calculated by equation 1. The function OOPS() is used for OOPS! tools [29] and shows the working of module 2 of phase 2. It returns an integer value by subtracting 1 from the pitfall rate (A). The function OCE() shows the ontology code effectiveness by calculating the size of the ontology (excluding annotation properties) [15] and shows the working of module 3

Algorithm for InFra OE

Input:
Expert E, KB, number of ontologies L
Function- KR(), OOPS(), OCE(), LBE ()
Integer- OE_1, OE_2, OE_3, OE_4, OE, KR, OOPS, LE, OE, KRR, FKRR, Ano, NoAno, Cov_S, Cov_C, Cov_R, Cov_{CP}, TPi, TP, N, Le, Se, Co, St, TLE, La, Ri, In, CO, Cl, Com, Ac, Re, Con, w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8
Boolean- $LExp_i$, Nl, Sb, Co, Re, Cp
Integer- TP=0; *Range*- {0.25, 0.50, 0.75, 1}

Processing:

$L \leftarrow Pick\ ontology\ from\ the\ KB$ // L is the total number of ontologies stored in the KB
For ($Li = 1;\ Li \leq L; Li++$) {
{
$OE_1 \leftarrow KR$ ()
$OE_2 \leftarrow OOPS$
$OE_3 \leftarrow OCE$
$OE_4 \leftarrow LBE$
}
Int KR () // function KR() corresponding to 5 role of KR
{
For ($E = 1;\ E \leq Ex;\ E++$) // Ex is the total number of experts
{
$Cov_S \leftarrow mean\ of\ rating\ given\ to\ questions\ Q1, Q2, and\ Q3$
$Cov_C \leftarrow mean\ of\ rating\ given\ to\ questions\ Q4, Q5, and\ Q6$
$Cov_R \leftarrow mean\ of\ rating\ given\ to\ questions\ Q7$
$Cov_{Cp} \leftarrow mean\ of\ rating\ given\ to\ questions\ Q8, and\ Q9$

$$KRR \leftarrow \frac{\exp\{-0.44+0.03(Cov_S \times Sb)E+0.02(Cov_C \times Co)E+0.01(Cov_R \times Re)E+0.02(Cov_{Cp} \times Cp)E-0.66LExp_E-25\ (0.1 \times Nl)E\}}{1+\exp\{-0.44+0.03(Cov_S \times Sb)E+0.02(Cov_C \times Co)E+0.01(Cov_R \times Re)E+0.02(Cov_{Cp} \times Cp)E-0.66LExp_E-25\ (0.1 \times Nl)E\}}$$

} FKRR ← FKRR + KRR
return FKRR }

Int OOPS () // function OOPS() corresponding to OOPS! tool
{
run OOPs! tool on the selected ontology Li
Pitfall ← calculate total number of cases of obtained pitfalls
{
For ($Pi = 1;\ Pi \leq Pitfall;\ Pi++$)
{
TPi ← cases of Pi
TP ← TP + TPi
}
$Ano \leftarrow \frac{TP}{N}$ // N is the total number of tuples
NoAno ← 1- Ano
} return NoAno }

Int OCE() ← calculate size of ontology, excluding annotation properties // function OCE() corresponding ontology's size
Int LBE () {
For ($E = 1;\ E \leq Ex;\ E++$) // Ex is the total number of experts
{
$Le \leftarrow w_1 \times La + w_2 \times Ri$
$Se \leftarrow w_3 \times In + w_4 \times CO + w_5 \times Cl$
$Con \leftarrow w_6 \times Com + w_7 \times Ac + w_8 \times Re$
$St \leftarrow check\ semantic\ relations\ between\ words$
$LE \leftarrow \frac{Le + Se + Co + St}{4}$
} return LBE} }

Output:
$OE \leftarrow \frac{OE_1 + OE_2 + OE_3 + OE_4}{4}$ // OE is the value of evaluation results that lies between $0 \leq OE \leq 1$

of phase 2. This function also has an integer value. The last function LBE() evaluates the ontology with respect to the four layers of ontology and uses different formulas at different layers like the syntactic approach is used at the lexicon layer, wordnet is utilized at the structure layer, etc. This function shows the working of module 3 of phase 2. The expert involvement occurs in two modules (module 1: evaluation based on 5 roles of knowledge representation; module 4: layer-based ontology evaluation), and they assign the value to the various parameters ranging {0.25, 0.50, 0.75, 1}. This range is defined by Bandeira et al. [4] after experimentation (we have mapped the range provided by Bandeira et al. [4] on a 0 to 1 scale for consistency purpose). The final value of evaluation is determined by taking the mean value of the four modules.

Evaluation of InFra_OE Framework: For the evaluation of InFra_OE framework, we have taken four Covid-19 Ontologies namely CODO, COKPME, COVID19, and LONGCOVID [26]. The details of these ontologies are mentioned below-

1. An Ontology for Collection and Analysis of COviD-19 Data (CODO): The CODO ontology is a data model that publishes Covid-19 data on the web as a knowledge graph.
2. The CODO aims to show the patient data and cases of Covid-19. The latest version of COVID19 was released in Sept 2020.
3. COKPME: This ontology is used to analyse the precautionary measures that help in controlling the spread of Covid-19. COKPME ontology is able to handle the various competence questions. The latest version of COKPME was released in Sept 2021.
4. COVID-19 Surveillance Ontology (COVID19): This ontology supports surveillance activities and is designed as an application ontology for the Covid-19 pandemic. The developed COVID-19 surveillance ontology ensures transparency and consistency. The latest version of COVID19 was released in May 2020.
5. Long Covid Phenotype Ontology (LONGCOVID): It is RCGP RSC Long Covid Phenotype ontology. The latest version of LONGCOVID was released in Oct 2021.

We have examined the pitfalls of these ontologies by OOPS! tool and the obtained results are mentioned in Table 3. The numbers (e.g. 1, 2, 4,..) that are contained in Table 3 denotes the total number of cases in accordance with the given pitfalls, like CODO ontology contain 1 case of pitfall P04. The sign × indicates no pitfall case is available in the respective ontology.

Table 3: Obtained pitfalls of covid-19 ontologies

Ontologies → Pitfalls ↓	CODO	COKPME	COVID19	LONG COVID
Minor Pitfalls				
P04	1	2	4	1
P07	×	×	×	×
P08	58	14	×	12
P13	38	15	×	×
P20	3	×	×	×
P21	×	1	×	×
P22	1	1	×	×
P32		×	×	×
Important Pitfalls				
P10	1	1	1	×
P11	58	14	×	×
P24	4	×	×	×
P25	4	×	×	×
P30	2	×	×	×
P34	7	×	×	×
P38	1	×	1	1
P41	×	×	1	1
Critical Pitfalls				
P19	×	3	×	×

We have calculated the pitfall rate by using equation (2). The value for the parameters of five role of knowledge representation is mentioned Table 4 and these values are used according to equation (1) to calculate the quality of the ontology based on the five role of knowledge representation. The Figure 5 shows the comparison among the three ontology evaluation methods namely OOPS! tool, Five role of knowledge representation (KR), and the proposed InFra_Ont framework. The Figure 5 depicts that-

- OOPS! tool- The LONGCOVID ontology has 0.454 pitfall rate, which is the highest as compared to other Covid-19 ontologies and COVID19 ontology has 0.042 pitfall rate, which is the lowest as compared to other Covid-19 ontologies.
- Five Role of Knowledge Representation (KR)- The COKPME ontology has 0.3947 pitfall rate, which is the highest as compared to other Covid-19 ontologies and CODO ontology has 0.2950 pitfall rate, which is the lowest as compared to other Covid-19 ontologies.
- InFra_Ont Framework- The LONGCOVID ontology has 0.42405 pitfall rate, which is the highest as compared to other Covid-19 ontologies and CODO ontology has 0.1905 pitfall rate, which is the lowest as compared to other Covid-19 ontologies.

Table 4: Value for parameters of five role of knowledge representation

Ontologies → Parameters ↓	CODO	COKPME	COVID19	LONGCOVID
COV_S	0.75	0.75	0.833	0.833
COV_C	0.833	0.75	0.5	0.5
COV_R	0.75	0.5	0.75	0.75
COV_{CP}	0.875	0.75	0.625	0.625
$LExp_E$	0.75	0.75	0.75	0.75

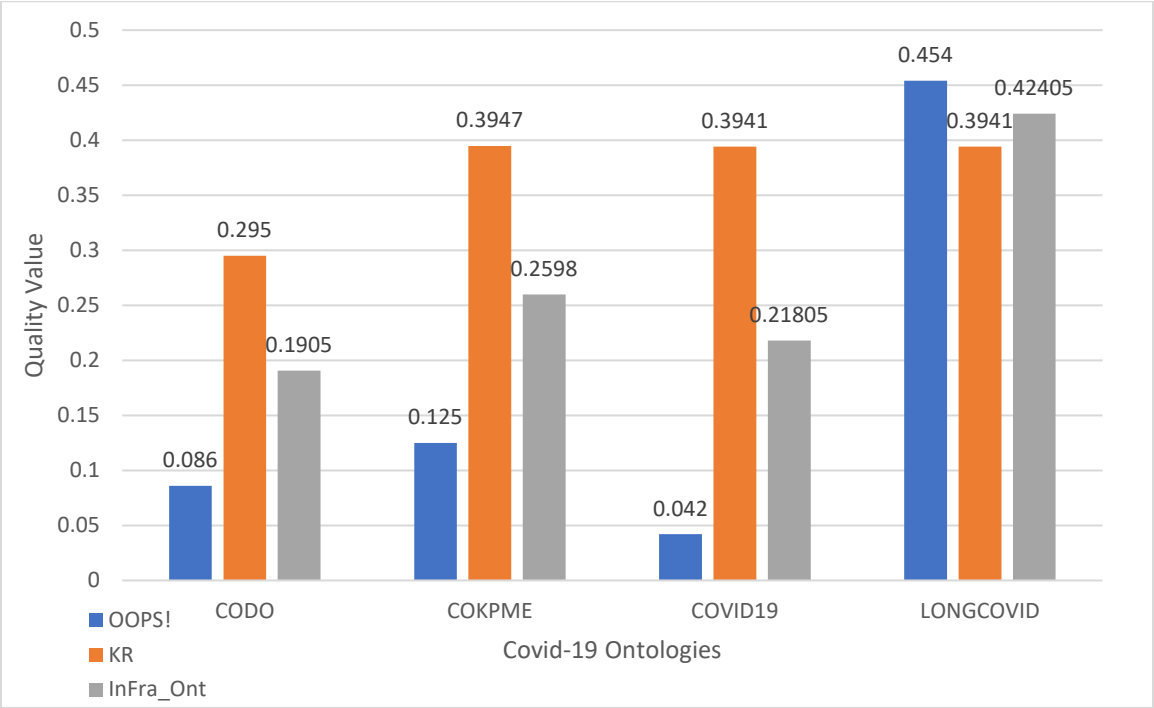


Figure 5: Ontology evaluation via OOPS! tool, KR, and InFra_Ont

The obtained results show that proposed INFra_Ont framework is better as compared to the existing frameworks as it successfully examines the good quality ontology among the available ontologies.

5 Conclusion and Future Research

Ontology provides a way to encode human intelligence so that machines can understand and make decisions by referring to this intelligence. For this reason, ontologies are used in every domain, and now, it becomes important to know the quality of the ontology. Ontology evaluation provides a set of methods and approaches that determine the quality of the ontology based on various criteria. The proposed integrated framework for ontology evaluation uses four well-known approaches to accommodate various criteria of evaluation. The proposed framework needs expert involvement in two approaches, namely evaluation based on the five roles of knowledge representation and evaluation based on the layer approach. The proposed algorithm shows the step-by-step execution of the InFra_OE and the evaluation of InFra_Ont framework shows that it is better as compared to the existing frameworks. The future work of this paper will be based on the evaluation of the proposed framework over different types of ontologies

References

[1] M. Amith, F. Manion, C. Liang, M. Harris, D. Wang, Y. He, and C. Tao, "OntoKeeper: Semiotic-Driven Ontology Evaluation Tool for Biomedical Ontologists." *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, pp. 1614-1617, December 2018.

[2] M. Amith and C. Tao, "Modulated Evaluation Metrics for Drug-Based Ontologies," *Journal of Biomedical Semantics*, 8(1):1-8, 2017.

[3] C. V. S. Avila, G. Maia, W. Franco, T. V. Rolim, A. D. O. da Rocha Franco, A. D. O., and V. M. P. Vidal, "OntoVal: A Tool for Ontology Evaluation by Domain Specialists," *ER Forum/Posters/Demos*, pp. 143-147, November 2019.

[4] J. Bandeira, I. I., Bittencourt, P., Espinheira, and S. Isotani, "FOCA: A Methodology for Ontology Evaluation," arXiv preprint arXiv:1612.03353, 2016.

[5] V. R. Basili and D. M. Weiss, "A Methodology for Collecting Valid Software Engineering Data," *IEEE Transactions on Software Engineering*, 6:728-738, 1984.

[6] A. B. Bouiadjra and S. M. Benslimane, "FOEval: Full Ontology Evaluation," *2011 7th International Conference on Natural Language Processing and Knowledge Engineering*, IEEE, pp. 464-468, November 2011.

[7] J. Brank, M. Grobelnik and D. Mladenic, "A Survey of Ontology Evaluation Techniques," *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*, Slovenia: Citeseer Ljubljana, pp. 166-170, October 2005.

[8] A. Burton-Jones, V. C. Storey, V. Sugumaran, and P. Ahluwalia, "A Semiotic Metrics Suite for Assessing the Quality of Ontologies," *Data & Knowledge Engineering*, *55*(1):84-102, 2005.

[9] M. Cristani and R. Cuel, "A Survey on Ontology Creation Methodologies," *International Journal on Semantic Web and Information Systems (IJSWIS)*, 1(2):49-69, 2005.

[10] R. Davis, H. Shrobe, and P. Szolovits, "What is a Knowledge Representation?" *AI magazine*, 14(1):17-17, 1993.

[11] N. C. Debnath, A. Patel, D. Mazumder, P. N. Manh, and N. H. Minh, "Evaluation of Covid-19 Ontologies through OntoMetrics and OOPS! Tools," International Conference on Expert Clouds and Applications (ICOECA), *Lecture Notes in Networks and Systems*, Springer (ISSN: 2367-3370), 2022.

[12] V. Devedzić, "Understanding Ontological Engineering," *Communications of the ACM*, 45(4), 136-144, 2002.

[13] R. Dividino, M. Romanelli, and D. Sonntag, "Semiotic-Based Ontology Evaluation Tool (S-OntoEval)," *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pp. 2687- 2692, May 2008.

[14] A. Duque-Ramos, J. T. Fernández-Breis, M. Iniesta, M. Dumontier, M. E. Aranguren, S. Schulz, and R. Stevens, "Evaluation of the OQuaRE Framework for Ontology Quality," *Expert Systems with Applications*, 40(7):2696-2703, 2013.

[15] H. Fujita and I. Zualkernan, "Evaluating Software Development Methodologies Based on Their Practices and Promises," *Proceedings of the Seventh Somet New Trends in Software Methodologies, Tools and Techniques 08*, *182*:14, 2008.

[16] N. Guarino and C. A. Welty, "An Overview of OntoClean" *Handbook on Ontologies*, pp. 151-171, 2004.

[17] A. Gyrard, M. Serrano, and G. A. Atemezing, "Semantic Web Methodologies, Best Practices and Ontology Engineering Applied to Internet of Things," *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, IEEE, pp. 412-417, December 2015.

[18] D. Kalibatiene, and O. Vasilecas, *Survey on Ontology Languages. International Conference on Business Informatics Research*, Springer, Berlin, Heidelberg, pp. 124-141, October 2011.

[19] D. D. Kehagias, I. Papadimitriou, J. Hois, D. Tzovaras, and J. Bateman, "A Methodological Approach for Ontology Evaluation and Refinement," *ASK-IT Final Conference. June. (Cit. on p.)* pp. 1-13, June 2008.

[20] S. Kim and S. G. Oh, "Extracting and Applying Evaluation Criteria for Ontology Quality Assessment," *Library Hi Tech*, Emerald Publishing, 37(3):338-354, 2019.

[21] S. Kolozali, T. Elsaleh, and P. M. Barnaghi, "A Validation Tool for the W3C SSN Ontology based Sensory Semantic Knowledge," *TC/SSN@ ISWC*, pp. 83-88, October 2014.

[22] B. Lantow, "OntoMetrics: Putting Metrics into Use for Ontology Evaluation," *KEOD*, pp. 186-191, November 2016.

[23] M. McDaniel and V. C. Storey, "Evaluating Domain Ontologies: Clarification, Classification, and

Challenges," *ACM Computing Surveys (CSUR)*, 52(4):1-44, 2019.

[24] K. Munir, and M. S. Anjum, "The Use of Ontologies for Effective Knowledge Modelling and Information Retrieval," *Applied Computing and Informatics*, 14(2):116-126, 2018.

[25] A. Patel, and N. C. Debnath, "Development of the InBan_CIDO Ontology by Reusing the Concepts Along with Detecting Overlapping Information," *Inventive Computation and Information Technologies*, Springer, Singapore, pp. 349-359, 2022.

[26] A. Patel, N. C. Debnath, A. K. Mishra, and S. Jain, "Covid19-IBO: A Covid-19 Impact on Indian Banking Ontology Along with an Efficient Schema Matching Approach," *New Generation Computing*, *39*(3):647-676, 2021.

[27] A. Patel, N. C. Debnath, and P, K. Shukla, "SecureOnt: A Security Ontology for Establishing Data Provenance in Semantic Web, *Journal of Web Engineering*, 21(4):1347-1370, 2022.

[28] A. Patel, and S. Jain, "A Partition based Framework for Large Scale Ontology Matching," *Recent Patents on Engineering*, 14(3):488-501, 2020.

[29] M. Poveda-Villalón, A. Gómez-Pérez and M. C. Suárez-Figueroa, "Oops!(Ontology Pitfall Scanner!): An On-Line Tool for Ontology Evaluation," *International Journal on Semantic Web and Information Systems (IJSWIS)*, *10*(2):7-34, 2014.

[30] P. Q. Rashid, *Semantic Network and Frame Knowledge Representation Formalisms in Artificial Intelligence*, Doctoral Dissertation, Eastern Mediterranean University (EMU)-Do˘gu Akdeniz Universitesi (DA ¨ U)), 2015...

[31] G. R. Roldan-Molina, J. R. Mendez, I. Yevseyeva, and V. Basto-Fernandes, "Ontology Fixing by Using Software Engineering Technology," *Applied Sciences*, 10(18):6328, 2020.

[32] S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth, and B. Aleman-Meza, "OntoQA: Metric-Based Ontology Quality Analysis," IEEE ICDM 2005 Workshop on Knowledge Acquisition, 2005.

[33] T. Uddin Haider, "Natural Language Text to RDF Schema Conversion and OWL Mapping for an e-Recruitment Domain," *Applications of Advanced Computing in Systems*, Springer, Singapore, pp. 49-57, 2021.

[34] M. Poveda Villalon, A. Gómez-Pérez, and M. C. Suárez-Figueroa, "OOPS!(Ontologty Pitfalls Scanner!): An On-Line Tool for Ontology Evaluation," *International Journal on Semantic Web and Information Sysems (IJSWIS)*, 10(2):7-34, 2012.

[35] J. Yu, J. A. Thom, and A. Tam, "Requirements-Oriented Methodology for Evaluating Ontologies," *Information Systems*, 34(8):766-791, 2009.

Narayan C. Debnath is currently the Founding Dean of the School of Computing and Information Technology at Eastern International University (EIU), Vietnam. He is also serving as the Head of the Department of Software Engineering at EIU, Vietnam. Dr. Debnath has been the Director of the International Society for Computers and their Applications (ISCA), USA since 2014. Formerly, Dr. Debnath served as a Full Professor of Computer Science at Winona State University, Minnesota, USA for 28 years where he also served as the Chairperson of the Computer Science Department for 7 years. Dr. Debnath earned a Doctor of Science (D.Sc.) degree in Computer Science and also a Doctor of Philosophy (Ph.D.) degree in Physics. In the past, he served as the elected President and Vice President of ISCA, and has been a member of the ISCA Board of Directors since 2001. Professor Debnath has made significant contributions in teaching, research, and services across the academic and professional communities. Dr. Debnath is an author or co-author of over 500 publications in numerous refereed journals and conference proceedings in Computer Science, Information Science, Information Technology, System Sciences, Mathematics, and Electrical Engineering. Dr. Debnath is an editor of several books published by Springer, Elsevier, CRC Press, and Bentham Science Press on emerging fields of computing. He also served as a guest editor of the Journal of Computational Methods in Science and Engineering (JCMSE) published by the IOS Press, the Netherlands.

Archana Patel is working as a faculty of the Department of Software Engineering, School of Computing and Information Technology, Eastern International University, Binh Duong Province, Vietnam. She has completed her Postdoc from the Freie Universität Berlin, Berlin, Germany. She has filed a patent titled "Method and System for Creating Ontology of Knowledge Units in a Computing Environment" in Nov 2019. She has received Doctor of Philosophy (Ph.D.) in Computer Applications and PG degree both from the National Institute of Technology (NIT) Kurukshetra, India in 2020 and 2016, respectively. She has qualified GATE and UGC-NET/JRF exams in year 2017. Dr.

Patel has also contributed in research project funded by Defence Research and Development Organization (DRDO), for the period of two year and her contribution in teaching is also remarkable. Dr. Patel is an author or co-author of more than 30 publications in numerous referred journals and conference proceedings. She has been awarded best paper award (four times) in the international conferences. She has served as a reviewer in various reputed journal and conferences. Dr. Patel has received various awards for presentation of research work at various international conferences, teaching and research institutions. She has edited three books, served as a guest editor in three reputed journals and served as a keynote at various reputed international conferences. Her research interests are Ontological Engineering, Semantic Web, Big Data, Expert System and Knowledge Warehouse.

Debarshi Mazumder is presently a lecturer in School of Computing & Information Technology at Eastern International University, Binh Duong, Vietnam. He received his M. Tech from Jadavpur University, Kolkata, India in 2013, and B. Tech from West Bengal University of Technology, Kolkata, India in 2008. His areas of research interests are Image Processing, Pattern Recognition, Machine Learning, IoT.

Manh-Phuc Nguyen received his undergraduate from Eastern International University in 2016. He finished his master's degree from the University of Information Technology, Ho Chi Minh National University, Viet Nam. He has been a full-stack software developer for 5 years and turned to be a lecturer of Software Engineering at Eastern International University. His research interests include computer vision, ontology learning, and software engineering. Recently, his work is focused on methods of object recognition and object tracking in traffic context and detect traffic violations. He would like to build a computer vision-based system to improve safety on the road.

Minh Ngoc Ha has 20 years of experience in the software development field, with over 15 years as a leader and 10 years as a technical director for some software firms. He turned to be a lecturer in Software Engineering at Eastern International University for nearly 10 years. His research interests include Data Structures and Algorithms, Software Engineering Methods, and teaching methodology. He loves to share his knowledge, skills, and experience with thousands of CIT students to promote them in their careers.

Journal Submission

The International Journal of Computers and Their Applications is published four times a year with the purpose of providing a forum for state-of-the-art developments and research in the theory and design of computers, as well as current innovative activities in the applications of computers. In contrast to other journals, this journal focuses on emerging computer technologies with emphasis on the applicability to real world problems. Current areas of particular interest include, but are not limited to: architecture, networks, intelligent systems, parallel and distributed computing, software and information engineering, and computer applications (e.g., engineering, medicine, business, education, etc.). All papers are subject to peer review before selection.

A. Procedure for Submission of a Technical Paper for Consideration

1. Email your manuscript to the Editor-in-Chief, Dr. Ajay Bandi. Email: ajay@nwmissouri.edu.
2. Illustrations should be high quality (originals unnecessary).
3. Enclose a separate page (or include in the email message) the preferred author and address for correspondence. Also, please include email, telephone, and fax information should further contact be needed.
4. **Note**: Papers shorter than 10 pages long will be returned.

B. Manuscript Style:

1. **WORD DOCUMENT**: The text should be **double-spaced** (12 point or larger), **single column** and **single-sided** on 8.5 X 11 inch pages. Or it can be single spaced double column.

   LaTex DOCUMENT: The text is to be a double column (10 point font) in pdf format.
2. An informative abstract of 100-250 words should be provided.
3. At least 5 keywords following the abstract describing the paper topics.
4. References (alphabetized by first author) should appear at the end of the paper, as follows: author(s), first initials followed by last name, title in quotation marks, periodical, volume, inclusive page numbers, month and year.
5. The figures are to be integrated in the text after referenced in the text.

C. Submission of Accepted Manuscripts

1. The final complete paper (with abstract, figures, tables, and keywords) satisfying Section B above in **MS Word format** should be submitted to the Editor-in-Chief. If one wished to use LaTex, please see the corresponding LaTex template.
2. The submission may be on a CD/DVD or as an email attachment(s). **The following electronic files should be included:**
   - Paper text (required).
   - Bios (required for each author).
   - Author Photos are to be integrated into the text.
   - Figures, Tables, and Illustrations. These should be integrated into the paper text file.
3. Reminder: The authors photos and short bios should be integrated into the text at the end of the paper. All figures, tables, and illustrations should be integrated into the text after being mentioned in the text.
4. The final paper should be submitted in (a) pdf AND (b) either Word or LaTex. For those authors using LaTex, please follow the guidelines and template.
5. Authors are asked to sign an ISCA copyright form (http://www.isca-hq.org/j-copyright.htm), indicating that they are transferring the copyright to ISCA or declaring the work to be government-sponsored work in the public domain. Also, letters of permission for inclusion of non-original materials are required.

Publication Charges

After a manuscript has been accepted for publication, the contact author will be invoiced a publication charge of **$500.00 USD** to cover part of the cost of publication. For ISCA members, publication charges are **$400.00 USD** publication charges are required.

Revised 2020