A Graph Matching Algorithm to extend Wise Systems with Semantic

Abdelhafid Dahhani, Ilham Alloui, Sébastien Monnet, Flavien Vernier

▶ **To cite this version:**

Abdelhafid Dahhani, Ilham Alloui, Sébastien Monnet, Flavien Vernier. A Graph Matching Algorithm to extend Wise Systems with Semantic. 18th Conference on Computer Science and Intelligence Systems, Sep 2023, Warsaw (POLAND), Poland. pp.411-420, 10.15439/2023f4672 . hal-04419947

HAL Id: hal-04419947
https://hal.science/hal-04419947v1

Submitted on 31 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 411–420
DOI: 10.15439/2023F4672
ISSN 2300-5963 ACSIS, Vol. 35

A Graph Matching Algorithm to extend Wise Systems with Semantic

Abdelhafid Dahhani, Ilham Alloui
0000-0001-6314-662X
0000-0002-3713-0592
Université de Savoir Mont Blanc,
LISTIC laboratoty,
Polytech Annecy-Chambery,
5 Chem. de Bellevue, 74940 Annecy, France
Email: {abdelhafid.dahhani, ilham.alloui}@univ-smb.fr

Sébastien Monnet, Flavien Vernier
0000-0002-6036-3060
0000-0001-7684-6502
Université de Savoir Mont Blanc,
LISTIC laboratoty,
Polytech Annecy-Chambery,
5 Chem. de Bellevue, 74940 Annecy, France
Email: {sebastien.monnet, flavien.vernier}@univ-smb.fr

***Abstract*—Software technology has exponentially evolved leading to the development of intelligent applications using artificial intelligence models and techniques. Such development impacts all scientific and social fields: home automation, medicine, communication, etc. To make those new applications useful to a larger number of people, researchers are working on how to integrate artificial intelligence into real world while respecting the notion of calm technology. This paper fits in the context of the development of intelligent systems termed "wise systems" that aim at satisfying the calm technology requirement. Those systems are based on the concept of "Wise Object": a software entity – object, service, component, application, etc. – able to learn by itself how it is expected to behave and how it is used by a human or another software entity. During its learning process, a Wise Object constructs a graph that represents its behavior and the way it is used. A major weakness of Wise Objects is that the numerical information that they generate is mostly meaningless to humans. Therefore the objective of the work presented in this paper is to extend Wise Objects with semantic that enable them communicate with humans whose attention will consequently be less involved. In this paper, we address the issue of how to relate two different views using two state-based formalisms: State Transition Graph for views generated by the Wise Objects and Input Output Symbolic Transition System for conceptual views. Our proposal extends previous work done to extend the generated information with the conceptual knowledge using a matching algorithm founded on graph morphism. The first version of the algorithm has several limitations and constraints on the graphs that make it difficult to use in realistic cases. In this paper, we propose to generalize the algorithm and raise those restrictions. To illustrate the complete process, the construction of a sample graph matching on a home-automation system is considered.**

I. Introduction

TO ENABLE usability and accessibility of intelligent systems to a large number of people, researchers are working on how to integrate artificial intelligence (AI) into real world while respecting the notion of calm technology. Calm technology represented by Mark Weiser and John Seely Brown [1] in 1995, intends to lightly involve humans within the work process by requiring the smallest possible amount of their attention [2], minimizing therefore system intrusion into their life. Furthermore as software systems usage varies depending on users and time, they should be able to autonomously adapt to evolving such usages. To meet those requirements, we realised a software framework to develop intelligent systems termed "Wise Systems" (WS), based on the concept of "Wise Object" (WO) [3].

A WO is a software entity – object, service, component, application – able to learn by itself how it is expected to behave and how it is used by a human or another software entity. This is enabled by introspection and reflection mechanisms (more details in [4]). WOs compose a WS that may be considered as a multi-agent system [5] [6] where a WO is a self-learning agent that autonomously monitors its internal changes and that does not know the other WOs in the system. Thus, a WO informs the WS about its state changes, so that other WOs react accordingly.

Based on the behaviour of the WS and the internal monitoring of each WO, the collected monitoring data can feed a learning process able to determine usual and unusual behaviors (for instance). As the development of WSs is non-trivial, we developed an object based framework, known as the Wise Object Framework (WOF [4]) to help developers design, deploy and evolve WSs. Generally, knowledge in AI-enabled systems can be provided according to two approaches: (*i*) describing a priori the arrangement of activities to be performed by the system, or, (*ii*) letting the system acquire by its own the required knowledge using different learning mechanisms:

- *In the former approach*, ontologies or scenarios are usually used to describe the arrangement of activities to achieve a goal as in [7] and [8]. In [7], functional behavior as well as inter-operation of system entities are described a priori using state-diagrams. In [8], the authors go a step forward combing ontologies to design ambiant assisted living systems with an unsupervised learning before system deployment to create relevant scenarios. In those approaches, the end user is at the heart of the scenario creation process as described in [9] and [10].
- *In the second approach*, knowledge is provided by the AI-enabled system in representations and views not necessarily understandable by humansand to the distance

between the business domain and technological domain views [11].

According to the calm technology [12], the WO embedded AI fits in the second approach. By self-monitoring, a WO acquires raw knowledge (logs) about its behavior. Based on IBM's 4 state loop (MAPE-K) [13] [14] architecture, WOF provides a plugin system to add analyzers that produce new knowledge from existing knowledge (i.e., logs or knowledge produced by the analyzers). Different analyzers already exist like statistical, Markovian, etc. In this paper, we focus on the State Transition Graph (STG) analyzer, it analyzes logs and produces an STG. This graph is useful for a WO to know in which state it stands and to determine the consequences on its state when it performs an action (a method invocation in the object oriented paradigm of WOF). This graph can also be used by the WO to determine the sequence of methods to call to reach a specific state. As previously stated, and because it is built in a completely unsupervised way, this graph does not carry useful semantic, the states only carry numbers, that is a problem to communicate with end-users.

Although initiated in the 18th century with Euler's work on the famous problem of Königsberg bridges [15], graph theory remains a powerful tool for software-intensive system development. Graphs are present at the software design stage, such as Input/Output Labelled Transitions Systems (IOLTS) or Input/Output Symbolic Transitions Systems (IOSTS) which are often used to model the system behavior, especially to build systems based on oracle, controller synthesis [16] [17] or to test a system by executing the various possible behaviors of this system [18], as well as during the data collection phase, to connect data from different sources [19]. As these graphs are conceptual, dedicated to human to human communication, they bring semantic. Since the STG built at runtime by a WO is close to IOLTS, a WO can use knowledge from its design stage, the IOSTS for example, to attach semantic to its STG. These graphs are obtained by combining two research methods (quantitative and qualitative). On the machine side, behavioral data (i.e., logs) is autonomously collected by WOs which dynamically analyse it and build an STG. On the human side, a developer/expert expresses the expected behavior of a system in a conceptual model using a human vocabulary. A qualitative behavioral view is provided using an IOSTS.

Our proposal in this paper is to present a new matching algorithm between both types of graphs. As the problem was simplified in our previous work [20], it has many limitations, the strongest being the number of equivalent attributes/variables in STG/IOSTS, another limitation is the constraint on the existence of only one matching between an STG and an IOSTS. This new algorithm will extend more further the generated knowledge with conceptual knowledge, based on the graph morphism [21] [22]. Thus, many variables from both knowledge will be taken into account this time, resulting in many different matching. This provides the ability to make WSs' generated knowledge understandable by human and to enable human evaluation of WSs' outputs. Explicitly, the contribution presented in this paper attempts to relate both views: (a) a conceptual view relying on knowledge given by developers to either describe or control the system behavior, and, (b) behavior-related knowledge acquired during WS's learning process, this process is illustrated through a concrete example. Consequently, we are in the process of establishing a machine-human communication (MHC). In this way, we use two state-based formalisms:

- *STG* for representing behavior-related knowledge generated by the WSs,
- *IOSTS* for modelling conceptual views of developers/experts.

This paper is organised as follows: Section II presents the basic idea, describes the architectural overview and gives the definition of important terms. Section III presents STG and IOSTS formalisms, and illustrates them through examples. Finally, Section IV presents our graph matching algorithm, before Section VI concludes the paper.

II. Basic idea & architectural view

The basic idea underlying the WO concept is to give a software entity the core mechanisms for learning behaviors through introspection and analysis. Our aim is to go further by enabling software to execute MAPE-K loops [4]. On the top of this concept, we built the WOF [3] with design decisions mainly guided by reusability and genericity requirements: the framework should be maintainable and used in different application domains with different strategies (e.g., analysis approaches).

Seeking clarity, we borrowed some terms used for humans to refer to abilities a WO possesses. Awareness and wisdom both rely on knowledge. Inspired by [23], we give some definitions of those terms commonly used for humans and present those we chose for WOs.

Knowledge: refers to information, inference rules and information deduced from them, for instance: "Turning on a heater will cause temperature change".

Awareness: represents the ability to collect - ability to provide internal data - on itself by itself. For instance, it is when an entity/object/device collects information and data about capabilities *(what is intended to do)* and its use *(what it is asked to do)*. Capabilities are the services and functionalities the WO may render.

Wisdom: is the ability to analyse collected information and stored knowledge related to their capabilities and usage to output useful information for end users. It is worth noticing that a WO is highly aware, while the converse is false. Wisdom implies awareness, but awareness does not imply wisdom.

Semantic: is the meaning given to something so that it can be understood by humans as mentioned in the Cambridge dictionary. This definition also applies to objects/devices, as semantic is used to communicate with humans.

From the conceptual view, according to the target application, a WO may be considered as:

- a stand-alone software entity (object, component, etc.),
- a software avatar designed to be a proxy for a physical device (e.g., a heater, vacuum cleaner, light bulb),

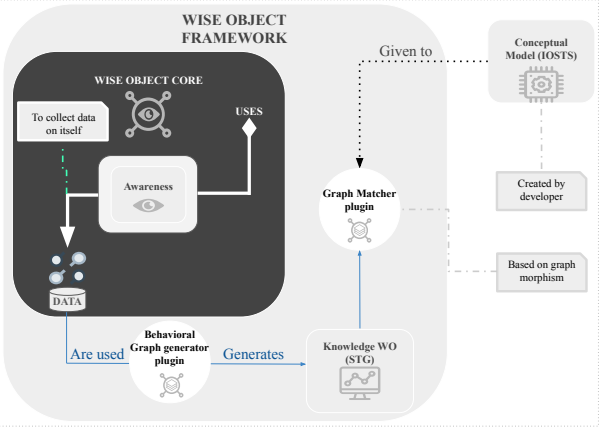

Fig. 1: Generic functional architecture of a WO in the WOF [20].

- a software avatar designed to be a proxy for an existing software entity (object, component, etc.).

A WO is characterised by its:

- autonomy: it is able to operate with no human intervention,
- adaptability: it changes its behavior when its environment evolves,
- ability to communicate (send its state changes and receive requests): with its environment according to a publish-subscribe paradigm.

Fig. 1 illustrates a partial view of a WO's functional architecture defined in the WOF. As depicted, the WO uses awareness to collect data on itself. It analyses those data thanks to the behavioral graph generator plugin (Also termed analyzer) and generates a behavioral graph represented by an STG (Section III-A). On the other hand, when designing an application, developers can provide a conceptual model describing/specifying the way they view the behavior of the system's entities associated to WOs. Such models are represented using IOSTS and contain the semantic given by developers to WOs (Section III-B). The IOSTS formalism is mostly known in simplifying system modelling by allowing symbolic representation of parameters and variable values instead of concrete data values enumeration [24].

The semantic carried by the IOSTS will be used by the graph matcher plugin to extend the STG using the algorithm proposed in this paper.

III. Complex Behavioral Models, Definitions and Illustrations

Modeling the behavior of a system is enabled by tools and languages that result in informal, semi-formal or formal representations: semi-formal notations like UML or more abstract behavior representations based on proven theories [25] like graph theory. In our case, STGs and IOSTSs respectively generated by WOs and provided by developers are used.

A. Definition of an STG

An STG is a directed graph where vertices represent the different states of an object and transitions represent the execution of its methods. Let us consider an object defined by its set of attributes A and its set of methods M. According to this information (A and M) on the object, the STG definition is given in Definition 1.

***Definition** 1:*

An STG is defined by the triplet $G(V, E, L)$ where:

- V is the set of vertices, with $|V| = n$ where each vertex represents a unique state of the object, and conversely, each state of the object is represented by a unique vertex. Therefore $v_i = v_j \Leftrightarrow i = j$ with $v_i, v_j \in V$ and $i, j \in [0, n[$.
- E is the set of directed edges where $\forall e \in E$, e is defined by the triplet $e = (v_i, v_j, m_k)$, such that $v_i, v_j \in V$ and $m_k \in M$. This triplet is called a transition labeled by m_k. The invocation of method m_k from state v_i switches the object to state v_j.
- L is a set of vertex labels where any label $l_i \in L$ is associated to v_i. A label l_i is the set of pairs $(att_j, value_{i,j})$ $\forall att_j \in A$, with $value_{i,j}$ the value of att_j in the state v_i and $Dom(att_j)$ the value domain of att_j, i.e., the set of $value_{i,j}$ for all i. By definition, 2 states v_i and v_j are different $v_i \neq v_j$, iff $\exists att_k \in A$, such that $value_{i,k} \neq value_{j,k}$. Conversely, if $\forall k \in [0, |A|[$ $value_{i,k} = value_{j,k}$, the states v_i and v_j are considered the same, i.e., $v_i = v_j$, thus $i = j$.

The STG must comply with certain constraints:

- The number of attributes is finite:

$$|A| \in \mathbb{N}^*.$$

- The domain $Dom(att_i)$ of each attribute $att_i \in A$ is bounded and discrete:

$$\begin{gathered}\forall att \in A, \min(Dom(att)) \neq -\infty, \\ \max(Dom(att)) \neq \infty, \\ |Dom(att)| \in \mathbb{N}^*.\end{gathered}$$

The matching algorithm we propose in Section IV takes as input an STG with a specific property we name exhaustiveness. Definition of "exhaustive STG" is given in Definition 2

***Definition** 2:* An exhaustive STG is an STG such that from each vertex v_i there exist $|M|$ transitions, each labeled by a method m_k in M:

$$\begin{gathered}\forall v_i \in V, \forall m_k \in M, \\ \exists v_j \in V | (v_i, v_j, m_k) \in E.\end{gathered} \tag{1}$$

It is worth noting that v_i and v_j may be different or same states ($v_i \neq v_j$ or $v_i = v_j$).

According to Definition 2, an exhaustive STG is deterministic, i.e., from any state, on any method invocation, the destination state is known. Moreover, the number of transitions $|E|$ in an exhaustive STG depends on the numbers of vertices $|V|$ and methods $|M|$ such that:

$$|V| \times |M| = |E|. \tag{2}$$

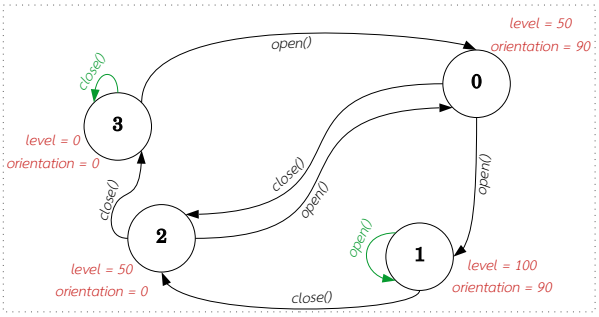

Fig. 2: An exhaustive STG presentation of a roller shutter.

As each vertex represents a state and a unique set of attribute values, an attribute is defined according a discrete and bounded domain and the set of attributes is naturally finite, the number of vertices $|V|$ is bounded:

$$\prod_{i=1}^{|A|} |Dom(att_i)| \geq |V|. \tag{3}$$

The inequality is due to the fact that some attribute value combinations may not be compatible. In other words the number of possible states is $\prod_{i=1}^{|A|} |Dom(att_i)|$, but all are not necessary reachable.

Fig. 2 illustrates an exhaustive STG of the behavior of a simplified roller shutter with adjustable slats. It is defined by the attributes "level" and "orientation" ($A = \{level, orientation\}$) and 2 methods "open" and "close" ($M = \{open(), close()\}$). The methods "open" and "close" respectively increase and decrease the level by 50, the slats orientation is adjusted automatically to have values 0 or 90. This STG was automatically generated and State 0 corresponds to the first discovered state, State 1 the second, etc., consequently, except the methods that give semantic to the transitions, the states have no semantic. According to the domains of the level and orientation, and Property 3, the STG has 4 reachable states, and according to Property 2 the number of edges is 8. Point out that State 0 has nothing to do with the initial values of the roller shutter, they correspond to the first state found during the automatic generation of the STG.

Let us note that a WO, never stores the whole STG due to an evident combinatorial explosion. Only a useful sub-graph is stored and mechanisms for forgetting sub-parts that are no longer useful are implemented. These memory problems are out of the scope of this paper, and we consider here only exhaustive STG to highlight the algorithm.

B. Definition of an IOSTS

An IOSTS is a directed graph whose vertices, called localities represent different states of the system (in our case, the system is a software object) and whose edges are transitions. The localities are connected by transitions triggered by actions. An IOSTS allows the definition of an infinite state transition system in a finite way, unlike an STG. In the literature, IOSTS are used to verify, test and control systems. Verification and testing are formal techniques for validating and comparing two views of a system while control is used to constrain the system behavior [16] [26]. The definition of IOSTS given in Definition 3, is taken from [27] [16] and especially from the use case given in [24].

***Definition** 3:* An IOSTS is a sixfold $\langle D, \Theta, Q, q_0, \Sigma, T \rangle$ such as:

- D is a finite set of typed data consisting of two disjoint sets of: variables X and action parameters P. Let an element $d \in D$, $Dom(d)$ determines the value domain of d.
- Θ an initial condition expressed as a predicate on variables X.
- Q is a non-empty finite set of localities with $q_0 \in Q$ the initial locality. A locality q is a set of states such that statesOf$(q) \subseteq Dom(X)$, with $Dom(X)$ the cartesian product of the domains $Dom(x)$ of each $x \in X$:
$$Dom(X) = \prod_{x \in X} Dom(x). \tag{4}$$
Let us note that a state is defined by a unique tuple of values for the whole variables.
- Σ is the alphabet, a finite, non-empty set of actions. It consists of the disjoint union of the set $\Sigma^?$ of input actions, the set $\Sigma^!$ of output actions, and the set $\Sigma^\mathcal{T}$ of internal actions. For each action a in Σ, its signature $sig(a) = \langle p_1, \ldots, p_k \rangle | p_i \in P$ is a tuple of parameters. The signature of internal actions is always an empty tuple.
- T is a finite set of transitions, such that each transition is a tuple $t = \langle q_o, a, G, A, q_d \rangle$ defined by:
  - a locality $q_o \in Q$, called the origin of the transition,
  - an action $a \in \Sigma$, called the action of the transition,
  - a boolean expression G on $X \cup sig(a)$ related to the variables and the parameters of the action, called the transition guard, transition guards allows us to distinguish transitions that have same origin and action but disjoint conditions to their triggering,
  - an assignment of the set of variables, of the form $(x := A^x)_{x \in X}$ such that for each $x \in X$, A^x is an expression on $X \cup sig(a)$, it defines the change of variable values during the transition,
  - a locality q_d, called the transition destination.

According to this definition, each variable has a subdomain in each locality. Thus, let us define the function $dom(q, x)$ that returns the definition domain of the variable $x \in X$ in the locality $q \in Q$, consequently $dom(q, x) \subseteq Dom(x)$. By extension, $dom(q, X)$ is the cartesian product of domains of all x in q:

$$\begin{gathered} dom(q, X) = \textstyle\prod_{x \in X} dom(q, x), \\ dom(q, X) \subseteq Dom(X). \end{gathered} \tag{5}$$

Fig. 3 illustrates the IOSTS given by a developer to control a roller shutter with adjustable slats. This IOSTS expresses that the roller shutter expects an input $up?/down? \in \Sigma^?$ carrying the parameter $step \in]0, 100]$, the relative elevation to increase or decrease the shutter level. In addition, the roller shutter slats can be rotated between 0 and 90 degrees during the elevation.

Let us mention that the roller shutter used in this example adapts automatically its $angle$, while the elevation is between 0 and 100 steps. Thus, end-users only control the elevation (height). The IOSTS contains two localities:

- The locality where the system is closed (i.e., both variables are equal to 0, see Equation 6). In this locality, if the system receives the $up?(step)$ command, the transition will be made from the $Closed$ to $Open$ locality by increasing the value of the $height$ variable by $step$. If the system receives the $down?(step)$ action, it will perform no operation (NOP).
- The locality where the system is open (i.e., one of the variables is different from 0, see Equation 7). In this locality, if the system receives the action $up?(step)$, the transition will be reflexive from $Open$ to itself and will compute the value of the variable $height$ by executing this assignment $height := min(height + step, 100)$, the shutter elevation cannot be increased more than the maximum of elevation. If it receives the $down?(step)$ action and the action closes the shutter less than it is open ($step < height$), $height$ is decreased by $step$, otherwise the transition will be from the locality $Open$ to the locality $Closed$ by assigning 0 to the variable $height$.

As illustrated in Fig. 3, and according to the Definition 3, the IOSTS is composed of:

- $Q = \{Closed, Open\}$, the set of localities.
- $X = \{height, angle\}$, the set of variables.
- $P = \{step\}$, the set of parameters.
- $\Sigma = \{up?, down?\}$, the set of actions where the signatures of the actions are $sig(up?) = sig(down?) = \langle step \rangle$.
- $Dom(height) = [0, 100]$, $Dom(angle) = [0, 90]$ and $Dom(step) =]0, 100]$ are respectively the domains of $height$, $angle$ and $step$.
- According to Equation 4, $Dom(X)$ is:

$$Dom(X) = [0, 100] \times [0, 90].$$

- The states of closed locality, defined by Equation 5, are:

$$\begin{gathered} statesOf(Closed) = \{(0, 0)\}, \\ \{(0, 0)\} \subset Dom(X). \end{gathered} \tag{6}$$

- The states of open locality are defined as follows:

$$\begin{gathered} statesOf(Open) = Dom(X) \backslash statesOf(Closed), \\ Dom(X) \backslash statesOf(Closed) \subset Dom(X). \end{gathered} \tag{7}$$

Thus:

$$statesOf(Closed) \cap statesOf(Open) = \emptyset. \tag{8}$$

As IOSTS is classical reference modeling formalism for model-based testing of reactive systems [28], it provide a convenient abstraction of the behaviors of such systems, which are beneficial and playing an important role in the matching algorithm.

IV. Graph matching algorithm

This section introduces our algorithm that relates the generated STG to developers' semantic expressed in an IOSTS formalism. The generated STG in Fig. 2 is composed of states automatically labelled by the WO: 0, 1, 2 and 3 and the localities of the IOSTS are labelled "Open" and "Closed". As both represent the same roller shutter, the main challenge is how to match states 0, 1, 2 and 3 to the localities of the IOSTS, in other words, which attribute of the STG corresponds to which variable of the IOSTS.

A. Matching constraints

The STG and IOSTS must meet certain criteria to correctly apply the matching algorithm.

1) Considering that we have in the set of attributes A a non empty subset called $A_e \neq \emptyset$ and in the set of variables X a non empty subset called $X_e \neq \emptyset$. Furthermore, let us consider $\mathcal{R}$ the binary relation of A_e in X_e, which is a bijection ($\rightarrowtail\!\!\!\!\!\rightarrow$) [29]:
   - each member of A_e must be linked exactly to one element of X_e,
   - each element of X_e must be linked exactly to one member of A_e.

$$\begin{gathered} \exists! A_e \subseteq A, \exists! X_e \subseteq X | A_e \rightarrowtail\!\!\!\!\!\rightarrow X_e \\ \Leftrightarrow \\ (A_e, X_e), \end{gathered} \tag{9}$$

   where (A_e, X_e) means that both A_e and X_e represent the same information. The matching solution is given by $(A_e, X_e, \mathcal{R})$. Therefore, any couple att_e, x_e such that $att_e \mathcal{R} x_e$ will be featured by (att_e, x_e) as they represent the same information, thus:

$$Dom(att_e) \subseteq Dom(x_e). \tag{10}$$

   Let us note that $Dom(att_e)$ is a subset of $Dom(x_e)$ due to the fact that $Dom(x_e)$ can be defined as a continuous domain and $Dom(att_e)$ is defined as a discrete and bounded set of values. Furthermore, from Equation 10 we deduce the following:
   - In case of $Dom(att_e) \subset \mathbb{R}$:

$$\begin{gathered} (att_e, x_e) \\ \Leftrightarrow \\ min(Dom(att_e)) \geq min(Dom(x_e)) \\ \wedge \\ max(Dom(att_e)) \leq max(Dom(x_e)). \end{gathered}$$

   - In case of $Dom(att_e) \subset \mathbb{S} \mid \mathbb{S}$ is the set of strings:

$$\begin{gathered} (att_e, x_e) \\ \Leftrightarrow \\ \forall value_i \in Dom(att_e), value_i \in Dom(x_e). \end{gathered}$$

2) Every locality in the IOSTS must be unique taking into account just the set variables X_e. Thus, the domains of X_e in the different localities in the IOSTS are disjoint:

$$\begin{gathered} \forall q, q' \in Q \mid q \neq q' \\ \Leftrightarrow \\ dom(q, X_e) \cap dom(q', X_e) = \emptyset. \end{gathered}$$

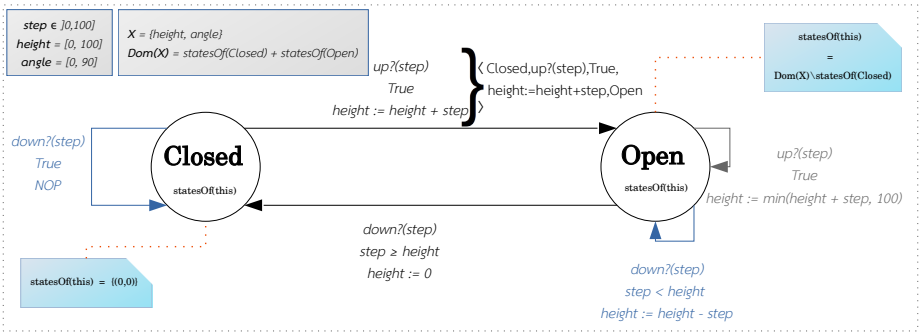

Fig. 3: An IOSTS representation of a roller shutter.

B. Algorithm

The matching algorithm will automatically run through several steps summarized in Fig. 4. The algorithm will receive two types of knowledge representation (STG and IOSTS). It is divided into three parts. The former produces all possible attribute-variable matching pairs P. The result will be used in the second part to build all longest combinations $\mathcal{P}_m$, which will be used to construct the matching between states and localities in the third part.

As validation of the algorithm is NP-complete, reducing the matching cost is planned as future work through the use of ontology and the matrix structure of graphs.

Let us detail the algorithm:

- As a starting point, P is the set that contains all potential equivalent attribute-variable pairs according to their domains:

$$P = \{(att, x) \mid att \in A, x \in X, Dom(att) \subseteq Dom(x)\}.$$

- $\mathcal{P}(P)$ is the power set of P [30], it contains all subsets c of P:

$$\forall c \subseteq P, c \in \mathcal{P}(P),$$

thus, $|\mathcal{P}(P)| = 2^{|P|}$. As A_e and X_e are not empty, the empty set can be removed from $\mathcal{P}(P)$:

$$\mathcal{P}_{\emptyset} = \mathcal{P}(P) \backslash \{\emptyset\}.$$

- $\mathcal{P}_v$ is all valid combinations c of pairs attribute-variable in $\mathcal{P}_{\emptyset}$. A combination c_i is valid if and only if it represents a bijection between its attributes and variables.

$$\mathcal{P}_v = \{c \mid c \in \mathcal{P}_{\emptyset}, \forall (att_i, x_j) \in c, \\ (att_i, x_k) \notin c \wedge (att_l, x_j) \notin c\},$$

with $i \neq l$ and $j \neq k$.

- $\mathcal{P}_m$ is all maximized combinations[1] c_i of pairs attribute-variable in $\mathcal{P}_v$:

$$\mathcal{P}_m = \{c \mid c \in \mathcal{P}_v, \forall c_j \in \mathcal{P}_v, c \not\subset c_j\}. \quad (11)$$

[1]The longest combinations such that any subset of a combination does not exist in the set of combinations.

$\mathcal{P}_m$ stores the maximized combinations according to the definition domain of attributes and variables.

As for each vertex it exists a unique locality such that for any couple attribute-variable of a combination, the values of the attribute are included in the domain of the locality, we keep from $\mathcal{P}_m$ only the combinations that satisfy such property. Therefore, $\mathcal{P}_{VQ}$ stores the possibles combinations c_i that correspond to a valid matching between vertices V and localities Q.

$$\mathcal{P}_{VQ} = \{c \mid c \in \mathcal{P}_m, \exists! v \in V, \exists! q \in Q, \forall (att, x) \in c, \\ dom(v, att) \subseteq dom(q, x)\}. \quad (12)$$

From $\mathcal{P}_{VQ}$, all the possible matching $\mathcal{M}_{VQ}$ that stores the sets of vertex-locality couples can be deduced:

$$\mathcal{M}_{VQ} = \{m_i \mid \forall c_i \in \mathcal{P}_{VQ}, \exists! v \in V, \exists! q \in Q, \\ \forall (att, x) \in c_i, dom(v, att) \subseteq dom(q, x), \\ (v, q) \in m_i\}. \quad (13)$$

These matching $\mathcal{M}_{VQ}$ are valid regarding the couples attribute att variable x and the couples vertex v locality q. Since the matching algorithm is based on the graph morphism, it needs to respect the structure of the matched graphs [31]. In our context, the images of the vertices of the STG in the IOSTS – the localities – must respect the *adjacency relations (neighboring)* present in the STG (i.e., the transitions). In other words, two adjacent vertices must match the same or two adjacent localities. Consider $\mathcal{S}$ is the surjective application of the STG in the IOSTS respectively between the vertices V and localities Q (see Equation 14), i.e., each vertex matches one locality and a locality is matched by at least one vertex. For any transition $(u, v) \in E$ of STG, then $(\mathcal{S}(u), \mathcal{S}(v)) \in T$ is a transition of the IOSTS. The STG $\rightarrow$ IOSTS matching is a surjective homomorphism, i.e., an epimorphism [31].

$$\begin{array}{rl} \mathcal{S}\colon STG & \rightarrow IOSTS, \\ V & \twoheadrightarrow Q = \mathcal{S}(V), \end{array} \quad (14)$$

implies:

$$\begin{array}{c} \mathcal{S}_E\colon E \rightarrow T, \\ \mathcal{S}_E((u, v)) = (\mathcal{S}(u), \mathcal{S}(v)), \\ (\mathcal{S}(u), \mathcal{S}(v)) \subset T. \end{array} \quad (15)$$

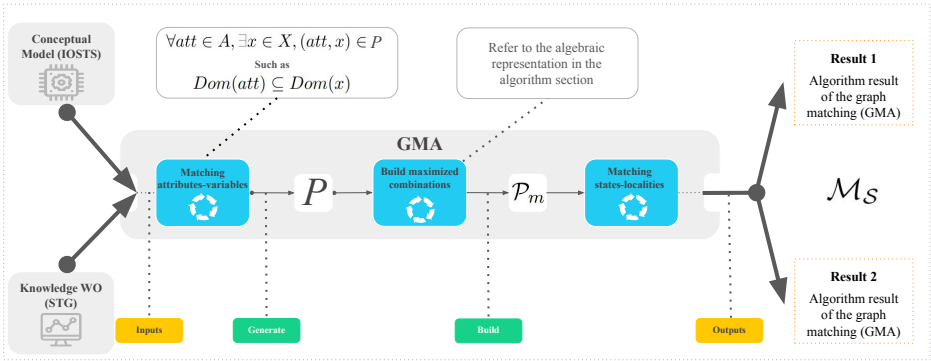

Fig. 4: Illustration of stepwise matching algorithm (outputs in Fig. 5 and Fig. 6)

.

According to Equation 15, if state v' is the neighbor of v and the locality q matches with v, v' must match with q or a neighbor of q:

$$\mathcal{M}_S = \{m \mid m \in \mathcal{M}_{VQ}, \forall (v,q) \in m, \forall v' \in neighbors(v), \exists q' \in neighbors(q) \cup \{q\}, (v', q') \in m\}. \tag{16}$$

C. Matching illustration

In the previous example, the STG in Fig. 2 is automatically generated by a WO and the IOSTS in Fig. 3 is provided by a developer. Both represent the same simplified roller shutter behavior. The behavior is simplified to highlight the algorithm, the implementation can deal with complex behaviors. The STG uses discrete values with a level of opening of 50% and a slats orientation of 90%, while the IOSTS use continuous intervals, without any constraint on the step that is a real value.

Fig. 5 and Fig. 6 illustrate all possible matching results of the API developed in the LISTIC laboratory of both (STG and IOSTS) graphs. Localities in the IOSTS are $Closed$ and $Open$, each contains variables with disjoint domains (see Equation 8), in our example, both variables $height$ and $angle$.

According to the constraints of the matching algorithm given in Section IV-A:

1) there is potential equivalent attributes/variables between the STG and the IOSTS, more precisely between the attributes “level, orientation” and the variables “height, angle”. According to Equation 10, the following pairs represent potential attributes/variables equivalences:

$$(level, height),$$
$$(orientation, angle),$$
$$(orientation, height),$$

2) the domains of both localities $Closed$ and $Open$ respect Equation 8. Thus, $Closed$ and $Open$ are disjoint.

On the STG side, there are four vertices, each one labeled with a set of attribute-value pairs $(att, value)$. In our case, the pair $(level, orientation)$ takes the values $[(50, 90), (100, 90), (50, 0), (0, 0)]$ respectively for

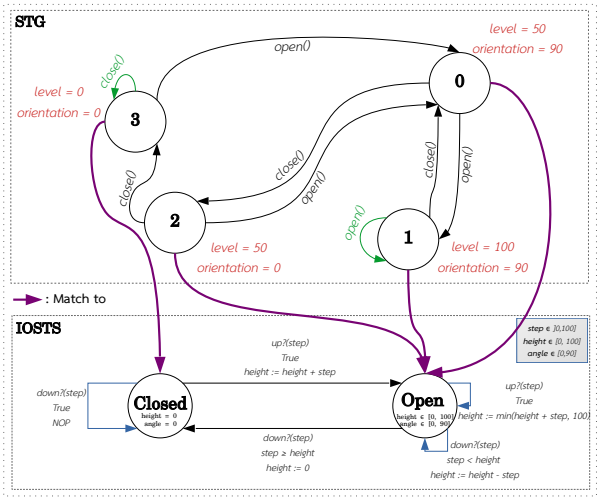

Fig. 5: Algorithm result of the graph matching $\mathcal{P}_m^1 = \{(level, height), (orientation, angle)\} \in \mathcal{P}_{VQ}$.

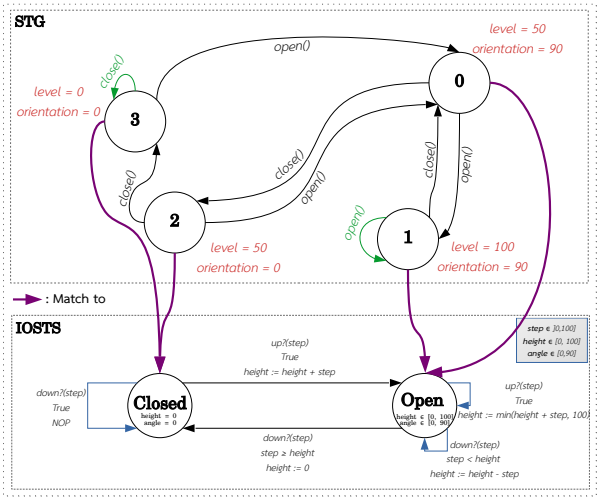

Fig. 6: Algorithm result of the graph matching $\mathcal{P}_m^2 = \{(orientation, height)\} \in \mathcal{P}_{VQ}$.

(v_0, v_1, v_2, v_3). Therefore, to establish a correspondence between the two graphs, for all combinations $\mathcal{P}_m(P)$, the definition domain of the pair $(level, orientation)$ in each vertex of the STG must be compared to the definition domain of the variables pair $(height, angle)$ in each locality of the IOSTS. This comparison gives $\mathcal{P}_{VQ}$, which consequently implies $\mathcal{M}_{\mathcal{S}}$. In detail, the following sets are obtained:

$$P = \{(level, height), (orientation, angle), (orientation, height)\}.$$

The powerset of P:

$$\begin{aligned}\mathcal{P}_{\emptyset}(P) = \{&\{(level, height)\}, \{(orientation, angle)\},\\ &\{(orientation, height)\},\\ &\{(level, height), (orientation, angle)\},\\ &\{(level, height), (orientation, height)\},\\ &\{(orientation, angle), (orientation, height)\},\\ &\{(level, height), (orientation, angle),\\ &(orientation, height)\}\}.\end{aligned}$$

Therefore:

$$\begin{aligned}\mathcal{P}_v = \{&\{(level, height)\}, \{(orientation, angle)\},\\ &\{(orientation, height)\},\\ &\{(level, height), (orientation, angle)\}\}.\end{aligned}$$

Thus:

$$\begin{aligned}\mathcal{P}_m = \{&\{(orientation, height)\},\\ &\{(level, height), (orientation, angle)\}\}.\end{aligned}$$

As $\mathcal{P}_{VQ} \equiv \mathcal{P}_m$ in this illustration, it contains two combinations, which gives two matches

$$\begin{aligned}\mathcal{M}_{VQ} = \{&\\ &\{(v_0, Open), (v_1, Open), (v_2, Open), (v_3, Closed)\},\\ &\{(v_0, Open), (v_1, Open), (v_2, Closed), (v_3, Closed)\}\\ \}.&\end{aligned}$$

Since both matching are surjective in this illustration, $\mathcal{M}_{VQ} \equiv \mathcal{M}_{\mathcal{S}}$.

This example provides two possible matches and the algorithm cannot determine, which one corresponds to (A_e, X_e). More information is required to determine the good matching. This information can be provided by the end user or, as we intend to do in future work, determined from the meaning of attributes, variables, methods and actions using an ontology.

V. Related work

For many years, graphs have been used in several fields to represent complex problems in a descriptive way (e.g., maps, relationships between people profiles, public transportation, scene analysis, chemistry, molecular biology, the quest for evolutionary conserved pathways thought protein-protein alignment, etc.). This has been done for various purposes: analysis, operation, knowledge modeling, pattern detection, etc. Although initiated in the 18th century with Euler's work on the problem of Königsberg bridges [15], graph theory remains a powerful tool for software-intensive system development and an effective way to represent objects as in [32]. Since then, several approaches of graph matching have been developed and the first formulation of the graph matching problem was proposed by [33] and dates back to 1979. Afterwards, several formulations appeared like convex-concave programming formulation, maximum common subgraph (MCS), the use of the Frobenius norm that uses the adjacency matrices of corresponding graphs to express the maximization or the minimization of the non-overlapping edges between two graphs, graph matching using dummy vertices that consist of finding a matching with the exception of some vertices in the data graph, which have no correspondence at all. In general, there exist two major formulations of the graph matching problem [34] [35]:

- *Exact Matching* is divided into two categories, (a) graph isomorphism, checks whether two graphs are the same. (b) subgraph isomorphism, checks whether the smallest graph is a subgraph of the biggest one. Both techniques are overly complex and rely on graph/subgraph isomorphism, whether or not they check the one-to-one or many-to-one matching.
- *Inexact Matching* is a term used where it is impossible to find an isomorphism between two graphs, and it comprises many approaches:
  - the maximum common subgraph, used in searching the similarity between the graphs to know how different they are instead of a binary answer [36]
  - least square formulation, used in the case of weighted graphs to search for a match that minimizes the total difference between all aligned edges through the use of the Frobenius norm for instance [36].
  - graph edit distance, used to find in a low cost the sequence of operations (i.e., deletion, insertion and substitution of vertices and edges) that transform one graph into another [37]. As this procedure is a hard combinatorial problem, another alternative called "beam search" is explained in details in [38]

In real applications, we often want to match graphs of different sizes, which results in new techniques and norms as depicted in [34]. Moreover, the problem is more extensive than one might imagine, as graphs are used to represent objects, images, regions in maps, also, many formalisms have emerged so far, such as the correspondence between different representations of knowledge as an STG and an IOSTS, which, to our knowledge, no paper has addressed. Until now, the most well-known operations on graphs is the comparison of two or more graph representations that requires many theoretical and complex concepts [21], like graph matching, which is a more constrained version of the graph isomorphism problem that is at the basis of our proposal. Finally, we mention that graph/sub-graph isomorphism and homomorphism are considered to be the most complex problems in graph matching, they are NP-complete. These problems have been studied in [39] [21] [40]. It is worth to mention that for certain

types of graphs under certain constraints, the complexity of the isomorphism has been proven to be of polynomial type with a huge cost [41].

The matching of two knowledge representations (STG - IOSTS) led us to two problems: exact matching and inexact matching. To understand the problems, we need to see the matching from both perspectives: machine (i.e., numerical and structural) and human (i.e., semantic). According to the machine, and since the matching preserves the structure and the transitions between both formalisms, the matching is always exact between "states" and "localities", which gives an epimorphism (Equations 14, 15). However, from the human perspective, in most real-life cases, there will be at most one exact matching according the semantic. As illustrated in the illustrations (Fig. 5 and Fig. 6), only one matching is exact. The exact matching problem is a great challenge, our work focuses on this problem by taking into account, in addition to the numerical perspective, the semantic perspective.

VI. Conclusion and future work

Our research work on software intelligent systems, namely WSs, tackles the issue of bringing closer knowledge generated by AI and human semantic. Indeed a major weakness of WOs composing a WS is that they generate numerical information mostly meaningless to humans. Our proposal is to extend knowledge issued by WOs (expressed in STGs) at runtime with knowledge (expressed in IOSTs) provided by developers at design time. Such extension is based on graph matching.

We have proposed in this paper an algorithm to face the problem of matching an STG and an IOSTS. The goal of this work is to extend WOs with the behavior semantic defined at software design time. From the end user's perspective, the algorithm provides the system with the ability to communicate with him using human semantic. From the developer's perspective, the resulted matching may help him discover errors or inconsistencies between the conceptual view and the system implementation. The results of the algorithm provide a set of valid matching according to the numerical and structural information stored in both behavioral graphs.

As illustrated in the paper, the algorithm provides a set of valid matching, however, it cannot determine the valid one from the end user perspective. Besides, the algorithm has neither an idea of the meaning of variables that represent the same information at different scales, nor of variable names. Thus, one of our future work will address this problem: How to take semantic into account in the matching problem? From the end user perspective, this problem is a semantic one, the natural solution is therefore to rely on ontologies. Approaches we are currently considering consist in semantic graph matching based on an ontology merged with the results presented in this paper.

In this respect, we have initiated a France-Canada innovation project with the University of Sherbrooke to investigate matching algorithms based on other semantic formalisms than IOSTSs, such as ontologies and scenario-based [8] [7]. The main idea is to bind the algorithm matching results with an ontology to provide a human level communication with users according to the context of the application. Moreover, this collaboration brings us the medical context as a new application domain, namely ambient assistance systems for elderly people.

Acknowledgment

This research was supported by French National Research Agency (ANR), AI Ph.D funding project.

References

[1] M. Weiser and J. S. Brown, "Designing calm technology," *POWERGRID JOURNAL*, vol. 1, 1996.

[2] A. Tugui, "Calm technologies in a multimedia world," *Ubiquity*, vol. 2004, 03 2004. doi: 10.1145/985616.985617

[3] I. Alloui, D. Esale, and F. Vernier, "Wise objects for calm technology," in *Proceedings of the 10th International Conference on Software Engineering and Applications - ICSOFT-EA, (ICSOFT 2015)*, INSTICC. SciTePress, 2015. doi: 10.5220/0005560104680471 pp. 468–471 – Section 2.

[4] S. Lejamble, I. Alloui, S. Monnet, and F. Vernier, "A new software architecture for the wise object framework: Multidimensional separation of concerns," in *Proceedings of the 17th International Conference on Software Technologies - ICSOFT,*, INSTICC. SciTePress, 2022. doi: 10.5220/0011355000003266 pp. 567–574.

[5] R. A. Flores-Mendez, "Towards a standardization of multi-agent system framework," *XRDS*, vol. 5, no. 4, p. 18–24, jun 1999. doi: 10.1145/331648.331659

[6] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 1–1, 2018. doi: 10.1109/ACCESS.2018.2831228

[7] D. Bonino and F. Corno, "Dogont - ontology modeling for intelligent domotic environments," in *The Semantic Web - ISWC 2008*, A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, and K. Thirunarayan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. doi: 10.1007/978-3-540-88564-1_51 pp. 790–803.

[8] H. K. Ngankam, H. Pigot, M. Frappier, O. Souza, H. Camila, and S. Giroux, "Formal specification for ambient assisted living scenarios," *UCAmI*, pp. 508–519, 11 2017. doi: 10.1007/978-3-319-67585-5_51

[9] J.-B. Woo and Y.-K. Lim, "User experience in do-it-yourself-style smart homes," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '15. New York, NY, USA: Association for Computing Machinery, 2015. doi: 10.1145/2750858.2806063 p. 779–790.

[10] R. Radziszewski, H. Ngankam, H. Pigot, V. Grégoire, D. Lorrain, and S. Giroux, "An ambient assisted living nighttime wandering system for elderly," in *Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services*, ser. iiWAS '16. New York, NY, USA: Association for Computing Machinery, 2016. doi: 10.1145/3011141.3011171 p. 368–374.

[11] R. S. Michalski, "A theory and methodology of inductive learning," *Artificial Intelligence*, vol. 20, no. 2, pp. 111–161, 1983. doi: 10.1016/0004-3702(83)90016-4

[12] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 66–75, January 1991. doi: 10.1145/329124.329126

[13] Y. Brun, G. D. M. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Muller, M. Pezzè, and M. Shaw, *Engineering Self-Adaptive Systems through Feedback Loops*. Springer Berlin Heidelberg, 2009, pp. 48–70.

[14] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, pp. 41 – 50, 02 2003. doi: 10.1109/MC.2003.1160055

[15] H. Sachs, M. Stiebitz, and R. Wilson, "An historical note: Euler's königsberg letters," *Journal of Graph Theory*, vol. 12, pp. 133–139, 10 2006. doi: 10.1002/jgt.3190120114

[16] C. Constant, T. Jéron, H. Marchand, and V. Rusu, "Integrating Formal Verification and Conformance Testing for Reactive Systems," *IEEE Transactions on Software Engineering*, vol. 33, no. 8, pp. 558–574, Aug. 2007. doi: 10.1109/TSE.2007.70707

[17] C. Camille, J. Thierry, M. Hervé, and R. Vlad, "Validation of Reactive Systems," in *Modeling and Verification of Real-TIME Systems - Formalisms and software Tools*, S, Merz, N, and Navet, Eds. Hermès Science, Jan. 2008, pp. 51–76. ISBN 978-1848210134

[18] I. Boudhiba, C. Gaston, L. G. Pascale, and V. Prevosto, "Input Output Symbolic Transition Systems Enriched by Program Calls and Contracts: a detailed example of vending machine," Laboratoire MAS - Centrale-Supelec, Research Report, 2015.

[19] J. Dörpinghaus, T. Hübenthal, and J. Faber, "A novel link prediction approach on clinical knowledge graphs utilizing graph structures," in *Proceedings of the 17th Conference on Computer Science and Intelligence Systems*, ser. Annals of Computer Science and Information Systems, vol. 30. IEEE, 2022. doi: 10.15439/2022F36 p. 43–52.

[20] A. Dahhani, I. Alloui, S. Monnet, and F. Vernier, "Towards a semantic model for wise systems - a graph matching algorithm," in *ADVCOMP 2022, The Sixteenth International Conference on Advanced Engineering Computing and Applications in Sciences*, S. Laura Garcia, Universitat Politecnica de Valencia, Ed., vol. 34, Nov. 2022. ISBN 2308-4499 pp. 27–34.

[21] M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Co., 1979. ISBN 0716710455

[22] V. A. Cicirello, "Survey of graph matching algorithms," Geometric and Intelligent Computing Laboratory, Drexel University, Technical Report, March 1999.

[23] T. Davenport and L. Prusak, *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press, 01 1998, vol. 1. ISBN 1578513014

[24] P. Moreaux, F. Sartor, and F. Vernier, "An effective approach for home services management," in *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2012. doi: 10.1109/PDP.2012.45 pp. 47–51.

[25] M. N. Nicolescu and M. J. Mataric, "Extending behavior-based systems capabilities using an abstract behavior representation," in *AAAI 2000*, North Falmouth, MA, 2000, pp. 27–34.

[26] V. Rusu, H. Marchand, V. Tschaen, T. Jéron, and B. Jeannet, "From safety verification to safety testing," in *Testing of Communicating Systems*, 03 2004. doi: 10.1007/978-3-540-24704-3_11. ISBN 978-3-540-21219-5 pp. 160–176.

[27] V. Rusu, H. Marchand, and T. Jéron, "Automatic verification and conformance testing for validating safety properties of reactive systems," in *Formal Methods 2005 (FM05)*, ser. Lecture Notes in Computer Science, vol. 3582. Newcastle, United Kingdom: Springer-Verlag, Jul. 2005. doi: 10.1007/11526841_14 pp. 189–204.

[28] C. Gaston, P. Le Gall, N. Rapin, and A. Touil, "Symbolic execution techniques for test purpose definition," in *Testing of Communicating Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. doi: 10.1007/11754008_1 pp. 1–18.

[29] M. Mashaal, *Bourbaki*, ser. Bourbaki : A secret society of mathematicians. American Mathematical Society, Jun. 2006. ISBN 9780821839676

[30] A. Salomaa, I. N. Sneddon, H. M. Stark, and J.-P. Kahane, *Theory of Automata : International Series of Monographs in Pure and Applied Mathematics*, ser. International series in pure and applied mathematics. - page.1-2. London : Elsevier Science, 2015., 1969-2015. ISBN 978-1483121970

[31] G. Hahn and C. Tardif, *Graph homomorphisms: structure and symmetry*. Dordrecht: Springer Netherlands, 1997, pp. 107–166. ISBN 978-94-015-8937-6

[32] M. Eshera and K.-S. Fu, "An image understanding system using attributed symbolic representation and inexact graph-matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 604–618, 1986. doi: 10.1109/TPAMI.1986.4767835

[33] W.-H. Tsai and K.-S. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern analysis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 12, pp. 757–768, 1979. doi: 10.1109/TSMC.1979.4310127

[34] M. Zaslavskiy, "Graph matching and its application in computer vision and bioinformatics," Theses, École Nationale Supérieure des Mines de Paris, Jan. 2010.

[35] E. Bengoetxea, "Inexact graph matching using estimation of distribution algorithms," Ph.D. dissertation, Ecole Nationale Supérieure des Télécommunications, Paris, France, Dec 2002.

[36] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, no. 1, p. 31–42, jan 1976. doi: 10.1145/321921.321925

[37] H. Bunke and G. Allermann, "Inexact graph matching for structural pattern recognition," *Pattern Recognition Letters*, vol. 1, no. 4, pp. 245–253, 1983. doi: 10.1016/0167-8655(83)90033-8

[38] M. Neuhaus, K. Riesen, and H. Bunke, "Fast suboptimal algorithms for the computation of graph edit distance," in *Structural, Syntactic, and Statistical Pattern Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. ISBN 978-3-540-37241-7 pp. 163–172.

[39] D. A. Basin, "A term equality problem equivalent to graph isomorphism," *Information Processing Letters*, vol. 51, no. 2, pp. 61–66, 1994. doi: 10.1016/0020-0190(94)00084-0

[40] M. A. Abdulrahim, *Parallel algorithms for labeled graph matching*. Colorado School of Mines1500 Illinois St. Golden, CO, 1998.

[41] J. E. Hopcroft and J. K. Wong, "Linear time algorithm for isomorphism of planar graphs (preliminary report)," in *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '74. Association for Computing Machinery, 1974. doi: https://dx.doi.org/10.1145/800119.803896 p. 172–184.