# Computational Verb Rule Bases

Tao Yang

***Abstract*— Different types of computational verb rule bases(verb rule bases, for short) are presented in this paper. The basic properties of verb rule bases are defined. The formal presentation of verb rule bases using Boolean matrices is given. The algorithms of merging and splitting verb rule bases using Boolean matrices are studied.** *Copyright © 2008 Yang's Scientific Research Institute, LLC. All rights reserved.*

***Index Terms*— Computational verb rule base, computational verb, Boolean matrix, rule base.**

## I. INTRODUCTION

COMPUTATIONAL verb rules are the basic building blocks for almost all engineering applications of computational verbs. In an engineering application, we usually need to model the input-output dynamics of a system. The inputs of the system can be different kinds of waveforms while the corresponding outputs can be different waveforms as well. In general, the system can have more than one input and more than one output. Therefore, to model the input-output relation of the system, we build a set of computational verb rules, of which the antecedents consist of multiple computational verbs and the consequences consist of multiple computational verbs. For each input or output, the dynamical behaviors are usually clustered into a group of computational verbs if there are any bifurcations along the parameter/phase space of the system, or, if the dynamics are quantitatively/qualitatively different along the parameter/phase space. In order to cover the input space and the output space completely, we need to construct a set of computational verb rules, of which many permutations of the computational verbs in antecedents and consequences are used. The function of each of these verb rules is to

1) cover a part of the entire parameter/phase space of the input waveforms (resulting in disjunctive antecedents);
2) cover the entire parameter/phase space of the input waveforms with its own emphasis (resulting in conjunctive antecedents);
3) cover a part of the entire parameter/phase space of the output waveforms (resulting in disjunctive consequences);
4) cover the entire parameter/phase space of the output waveforms with its own emphasis (resulting in conjunctive consequences).

Manuscript received January 6, 2008; revised August 08, 2008.

Tao Yang, Department of Electronic Engineering, Xiamen University, Xiamen 361005, P.R. China. Department of Cognitive Economics, Department of Electrical Engineering and Computer Sciences, Yang's Scientific Research Institute, 1303 East University Blvd., #20882, Tucson, Arizona 85719-0521, USA. Email: taoyang@xmu.edu.cn,taoyang@yangsky.com,taoyang@yangsky.us.

Publisher Item Identifier S 1542-5908(08)10308-6/$20.00



Therefore, there are four permutations of types of antecedents and consequences for constructing verb rules listed as follows.

1) Conjunctive antecedents with conjunctive consequences;
2) Conjunctive antecedents with disjunctive consequences;
3) Disjunctive antecedents with conjunctive consequences;
4) Disjunctive antecedents with disjunctive consequences.

When a set of verb rules; namely, a verb rule base is used to model the relations between dynamical inputs and outputs, there are at least two kinds of inter-rule relations

1) Conjunctive inter-rule relation. In this case, the relation between rules is in serial and each verb rule cover the entire input-output relation with emphasis.
2) Disjunctive inter-rule relation. In this case, the relation between rules if in parallel and each verb rule cover a part of input-output relation.

With all permutations of different types of antecedents, consequences and inter-rule relations, there are eight possible types of verb rule bases. Since in a real-life problem, it is difficult to find the crisp boundaries among qualitatively different dynamics, we usually design a computational verb to cover the entire parameter/phase space with emphasis. Therefore, the most commonly used verb rule bases are with conjunctive antecedents, conjunctive consequences and conjunctive inter-rule relations.

The organization of this paper is as follows. In Section II, the brief history of computational verb theory will be given. In Section III, different types of verb rule bases will be defined. In Section IV, the basic properties of verb rule bases will be presented. In Section V, formal presentation of verb rule bases in Boolean matrices will be given. In Section VI, the merging algorithm of verb rule bases will be designed based on the formal presentation of verb rule bases using Boolean matrices. In Section VII, the splitting algorithm of verb rule bases will be designed based on the formal presentation of verb rule bases using Boolean matrices. In Section VIII, some concluding remarks will be included.

## II. A BRIEF HISTORY OF COMPUTATIONAL VERB THEORY

As the first paradigm shift for solving engineering problems by using verbs, the computational verb theory[27] and physical linguistics[30], [47] have undergone a rapid growth since the birth of computational verb in the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley in 1997[13], [14]. The paradigm of implementing verbs in machines was coined as *computational verb theory*[27]. The building blocks of computational theory are *computational verbs*[22], [17], [15], [23], [28]. The relation between verbs and adverbs was mathematically defined in

[16]. The logic operations between verb statements were studied in [18]. The applications of verb logic to verb reasoning were addressed in [19] and further studied in [27]. A logic paradox was solved based on verb logic[24]. The mathematical concept of set was generalized into verb set in[21]. Similarly, for measurable attributes, the number systems can be generalized into verb numbers[25]. The applications of computational verbs to predictions were studied in [20]. In [29] fuzzy dynamic systems were used to model a special kind of computational verb that evolves in fuzzy spaces. The relation between computational verb theory and traditional linguistics was studied in [27], [30]. The theoretical basis of developing computational cognition from a unified theory of fuzzy and computational verb theory is the theory of the UNICOGSE that was studied in [30], [35]. The issues of simulating cognition using computational verbs were studied in [31]. In [54] the correlation between computational verbs was studied. A method of implementing feelings in machines was proposed based on grounded computational verbs and computational nouns in [37]. In [44] a theory of how to design stable computational verb controllers was given. In [38] the rule-wise linear computational verb systems and their applications to the design of stable computational verb controllers and chaos in computational verb systems were presented. In [42] the concept of computational verb entropy was used to construct computational verb decision tree for data-mining applications. In [41] the relation between computational verbs and fuzzy sets was studied by using computational verb collapses and computational verb extension principles. In [43] the distances and similarities of saturated computational verbs were defined as normalized measures of the distances and similarities between computational verbs. Based on saturated computational verbs, the verb distances and similarities are related to each other with a simple relation. The distances and similarities between verbs with different life spans can be defined based on saturated computational verbs as well. In [45] the methods of using computational verbs to cluster trajectories and curves were presented. To cluster a bank of trajectories into a few representative computational verbs is to discover knowledge from database of time series. We use cluster centers to represent complex waveforms at symbolic levels. In [11] computational verb controllers were used to control a chaotic circuit model known as Chua's circuit. Computational verb controllers were designed based on verb control rules for different dynamics of the region-wise linear model of the control plant. In [10] computational verb controllers were used to synchronize discrete-time chaotic systems known as Hénon maps. Different verb control rules are designed for synchronizing different kinds of dynamics. In [49], how can computational verb theory functions as the most essential building block of cognitive engineering and cognitive industries was addressed. Computational verb theory will play a critical important role in personalizing services in the next fifty years. In [46], [48] computational verb theory was used to design an accurate flame-detecting systems based on CCTV signal. In [52] the learning algorithms were presented for learning computational verb rules from training data. In [50] the structures and learning algorithms of computational verb neural networks were presented. In [53] the ambiguities of the states and dynamics of computational verbs were studied. In [51] the history and milestones in the first ten years of the studies of computational verb theory were given. In [3] a case study of modeling adverbs as modifiers of computational verbs was presented. In [12] computational verb rules were used to improve the training processes of neural networks.

The theory of computational verb has been taught in some university classrooms since 2005[1]. The latest active applications of computational verb theory are listed as follows.

1) Computational Verb Controllers. The applications of computational verbs to different kinds of control problems were studied on different occassions[26], [27]. For the advanced applications of computational verbs to control problems, a few papers reporting the latest advances had been published[33], [32], [44], [38], [55]. The design of computational verb controllers was also presented in a textbook in 2005[1].
2) Computational Verb Image Processing and Image Understanding. The recent results of image processing by using computational verbs can be found in[34]. The applications of computational verbs to image understanding can be found in [36]. The authors of [2] applied computational verb image processing to design the vision systems of RoboCup small-size robots.
3) Stock Market Modeling and Prediction based on computational verbs. The product of Cognitive Stock Charts[6] was based on the advanced modeling and computing reported in [39]. Computational verb theory was used to study the trends of stock markets known as Russell reconstruction patterns [40].

Computational verb theory has been successfully applied to many industrial and commercial products. Some of these products are listed as follows.

1) Visual Card Counters. The *YangSky-MAGIC* card counter[8], developed by Yang's Scientific Research Institute and Wuxi Xingcard Technology Co. Ltd., was the first visual card counter to use computational verb image processing technology to achieve high accuracy of card and paper board counting based on cheap webcams.
2) CCTV Automatic Driver Qualify Test System. The *DriveQfy* CCTV automatic driver qualify test system[9] was the first vehicle trajectory reconstruction and stop time measuring system using computational verb image processing technology.
3) Visual Flame Detecting System. The *FireEye* visual flame detecting system[4] was the first CCTV or webcam based flame detecting system, which works under color and black & white conditions, for surveillance and security monitoring system.

[1]Dr. G. Chen, EE 64152 - Introduction to Fuzzy Informatics and Intelligent Systems, Department of Electronic Engineering, City University of Hong Kong. Dr. Mahir Sabra, EELE 6306: Intelligent Control, Electrical and Computer Engineering Department, The Islamic University of Gaza. Dr. D. H. Guo, Artificial Intelligence, Department of Electronic Engineering, Xiamen University. Prof. T. Yang, Computational Methodologies in Intelligent Systems, Department of Electronic Engineering, Xiamen University.

4) Smart Pornographic Image and Video Detection Systems. The *PornSeer*[7] pornographic image and video detection systems are the first cognitive feature based smart porno detection and removal software.
5) Webcam Barcode Scanner. The *BarSeer*[5] webcam barcode scanner took advantage of the computational verb image processing to make the scan of barcode by using cheap webcam possible.
6) Cognitive Stock Charts. By applying computational verbs to the modeling of trends and cognitive behaviors of stock trading activities, cognitive stock charts can provide the traders with the "feelings" of stock markets by using simple and intuitive indexes.
7) TrafGo ITS SDK. Computational verbs were applied to model vehicle trajectories and dynamics of optical field and many other aspects of dynamics in complex environments for applications in intelligent transportation systems (ITS).

## III. Computational Verb Rule Bases

A typical engineering system consists of multiple inputs and multiple outputs as shown in Fig. 1. Since we usually observe these input/output signals over a certain period of time, the signals $x_i(t), i = 1, \ldots, m$; $y_j(t), j = 1, \ldots, o$ are usually represented as time series in digital systems or waveforms in analogue systems.

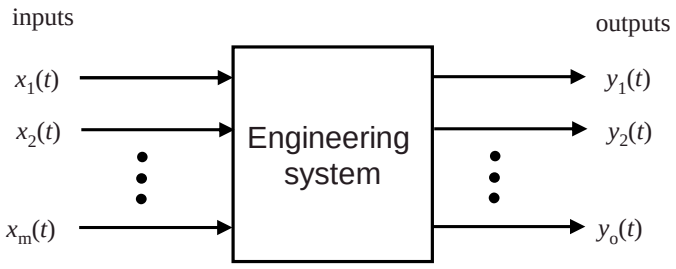


Fig. 1. Block diagram of the inputs and outputs of a typical engineering system.

To construct the model of the engineering system from the observations of its inputs and outputs is the problem of modeling. One way of modeling is to construct computational verb rules to define the relation between the dynamical behaviors of $x_i$'s and $y_j$'s. For each input $x_i(t)$, the dynamics can be lumped into some computational verbs. The same can be done to each output $y_j(t)$. Therefore, a typical verb rule for modeling an engineering system is usually of the form

$$\text{IF } [x_1 \ \mathsf{V}_1] \wedge \ldots \wedge [x_m \ \mathsf{V}_m], \text{THEN } [y_1 \ \widetilde{\mathsf{V}}_1] \wedge \ldots \wedge [y_o \ \widetilde{\mathsf{V}}_o] \quad (1)$$

where each $x_i$ and a prescribed verb constitute a verb phase in the antecedent while each $y_j$ and a prescribed verb constitute a verb phrase in the consequence.

Sine each $x_i$ and $y_j$ can associate with different verbs to form different verb phrases, there are many possible verb rules to choose from the permutations of all possible verbs associated to $x_i$'s and $y_j$'s in verb rules. Therefore, in a real-life engineering application, we usually face a set of verb rules of the same structure.

A *rule base* is a set of rules, therefore, a rule base is also known as an *algorithm*. If in a rule base,

- the relation between each rule in a verb rule base is conjunctive, then the rule base is called *conjunctive rule base*.
- the relation between each rule in a verb rule base is disjunctive, then the rule base is called *disjunctive rule base*.

Since in real-life problems, it is more difficult to find solutions to satisfy all rules at the same time in a rule base than solutions to satisfy one rule each time, disjunctive rule bases are more common than conjunctive rule base.

If the relation between all verb phrases in the antecedent of a computational verb rule is

- conjunctive, then the antecedent is called *conjunctive antecedent*.
- disjunctive, then the antecedent is called *disjunctive antecedent*.

Since in real-life problems, we usually expect all verb phrases in an antecedent to be satisfied at the same time, it is more commonly to use conjunctive antecedents than disjunctive ones.

Based on all permutations of types of antecedents and types of rule bases, we have the following four types of conjunctive-consequent rule bases.

1) Conjunctive antecedents in disjunctive(CAD) rule base as shown in Eq. (2).
2) Disjunctive antecedents in disjunctive(DAD) rule base as shown in Eq. (3).
3) Conjunctive antecedents in conjunctive(CAC) rule base as shown in Eq. (4).
4) Disjunctive antecedents in conjunctive(DAC) rule base as shown in Eq. (5).

CAD rule base (2) is the most commonly used one in real-life problems.

Let us assume that each of the rule bases (2) to (5) consists of $n$ computational verb rules, in each of which the antecedent consists of $m$ verb phrases and the consequent consists of $o$ verb phrases. The rule bases (2) to (5) are *multiple-input-and-multiple-output*(MIMO). We usually need to decompose a MIMO rule base into a *multiple-input-and-single-output*(MISO) rule set because the latter is easy to implement and analyze. As an example, for each rule in CAD rule base (2) we have

$$\begin{aligned}
& \text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{i1}] \wedge \ldots \wedge [y_o \ \widetilde{\mathsf{V}}_{io}]; \\
= \ & [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}] \rightarrow [y_1 \ \widetilde{\mathsf{V}}_{i1}] \wedge \ldots \wedge [y_o \ \widetilde{\mathsf{V}}_{io}] \\
= \ & \neg\{[x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}]\} \vee \{[y_1 \ \widetilde{\mathsf{V}}_{i1}] \wedge \ldots \wedge [y_o \ \widetilde{\mathsf{V}}_{io}]\} \\
= \ & \neg\{[x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}]\} \vee [y_1 \ \widetilde{\mathsf{V}}_{i1}] \\
& \wedge \ldots \wedge \neg\{[x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}]\} \vee [y_o \ \widetilde{\mathsf{V}}_{io}] \\
= \ & \{[x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}]\} \rightarrow [y_1 \ \widetilde{\mathsf{V}}_{i1}] \\
& \wedge \ldots \wedge \{[x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}]\} \rightarrow [y_o \ \widetilde{\mathsf{V}}_{io}] \\
= \ & \{\text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{i1}]\} \\
& \wedge \ldots \wedge \{\text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{io}]\}.
\end{aligned} \quad (6)$$

where "$\neg$" denotes logic negation.

$$
\begin{array}{l}
\text{IF } [x_1\ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{1m}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{11}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{1o}]; \\
\vee \\
\vdots \\
\vee \\
\text{IF } [x_1\ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{im}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{i1}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{io}]; \\
\vee \\
\vdots \\
\vee \\
\text{IF } [x_1\ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{nm}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{n1}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{no}].
\end{array} \tag{2}
$$

$$
\begin{array}{l}
\text{IF } [x_1\ \mathsf{V}_{11}] \vee \ldots \vee [x_m\ \mathsf{V}_{1m}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{11}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{1o}]; \\
\vee \\
\vdots \\
\vee \\
\text{IF } [x_1\ \mathsf{V}_{i1}] \vee \ldots \vee [x_m\ \mathsf{V}_{im}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{i1}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{io}]; \\
\vee \\
\vdots \\
\vee \\
\text{IF } [x_1\ \mathsf{V}_{n1}] \vee \ldots \vee [x_m\ \mathsf{V}_{nm}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{n1}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{no}].
\end{array} \tag{3}
$$

$$
\begin{array}{l}
\text{IF } [x_1\ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{1m}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{11}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{1o}]; \\
\wedge \\
\vdots \\
\wedge \\
\text{IF } [x_1\ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{im}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{i1}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{io}]; \\
\wedge \\
\vdots \\
\wedge \\
\text{IF } [x_1\ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{nm}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{n1}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{no}].
\end{array} \tag{4}
$$

$$
\begin{array}{l}
\text{IF } [x_1\ \mathsf{V}_{11}] \vee \ldots \vee [x_m\ \mathsf{V}_{1m}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{11}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{1o}]; \\
\wedge \\
\vdots \\
\wedge \\
\text{IF } [x_1\ \mathsf{V}_{i1}] \vee \ldots \vee [x_m\ \mathsf{V}_{im}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{i1}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{io}]; \\
\wedge \\
\vdots \\
\wedge \\
\text{IF } [x_1\ \mathsf{V}_{n1}] \vee \ldots \vee [x_m\ \mathsf{V}_{nm}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{n1}] \wedge \ldots \wedge [y_o\ \widetilde{\mathsf{V}}_{no}].
\end{array} \tag{5}
$$

Therefore, we can transform the MIMO verb rule base (2) into the following MISO verb rule base.

$$
\begin{array}{l}
\{\text{IF } [x_1 \ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{11}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{1o}]\} \\
\vee \\
\vdots \\
\vee \\
\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{i1}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{io}]\} \\
\vee \\
\vdots \\
\vee \\
\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{n1}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{no}]\}
\end{array} \tag{7}
$$

Similarly, the MIMO verb rule base (3) can be transformed into the following MISO verb rule base.

$$
\begin{array}{l}
\{\text{IF } [x_1 \ \mathsf{V}_{11}] \vee \ldots \vee [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{11}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{11}] \vee \ldots \vee [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{1o}]\} \\
\vee \\
\vdots \\
\vee \\
\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \vee \ldots \vee [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{i1}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \vee \ldots \vee [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{io}]\} \\
\vee \\
\vdots \\
\vee \\
\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \vee \ldots \vee [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{n1}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \vee \ldots \vee [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{no}]\}
\end{array} \tag{8}
$$

The MIMO verb rule base (4) can be transformed into the following MISO verb rule base.

$$
\begin{array}{l}
\{\text{IF } [x_1 \ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{11}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{1o}]\} \\
\wedge \\
\vdots \\
\wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{i1}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{io}]\} \\
\wedge \\
\vdots \\
\wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{n1}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{no}]\}
\end{array} \tag{9}
$$

The MIMO verb rule base (5) can be transformed into the following MISO verb rule base.

$$
\begin{array}{l}
\{\text{IF } [x_1 \ \mathsf{V}_{11}] \vee \ldots \vee [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{11}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{11}] \vee \ldots \vee [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{1o}]\} \\
\wedge \\
\vdots \\
\wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \vee \ldots \vee [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{i1}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \vee \ldots \vee [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{io}]\} \\
\wedge \\
\vdots \\
\wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \vee \ldots \vee [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{n1}]\} \\
\wedge \ldots \wedge \\
\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \vee \ldots \vee [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{no}]\}
\end{array} \tag{10}
$$

All consequences in rule bases (2) to (5) are conjunctive. If these conjunctive consequences are replaced by disjunctive consequences, then we have the following four types of rule bases.

1) Conjunctive antecedents in disjunctive(CAD) rule base as shown in Eq. (11).
2) Disjunctive antecedents in disjunctive(DAD) rule base as shown in Eq. (12).
3) Conjunctive antecedents in conjunctive(CAC) rule base as shown in Eq. (13).
4) Disjunctive antecedents in conjunctive(DAC) rule base as shown in Eq. (14).

$$
\begin{array}{l}
\text{IF } [x_1\ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{1m}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{11}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{1o}]; \\
\vee \\
\vdots \\
\vee \\
\text{IF } [x_1\ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{im}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{i1}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{io}]; \\
\vee \\
\vdots \\
\vee \\
\text{IF } [x_1\ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{nm}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{n1}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{no}].
\end{array} \tag{11}
$$

$$
\begin{array}{l}
\text{IF } [x_1\ \mathsf{V}_{11}] \vee \ldots \vee [x_m\ \mathsf{V}_{1m}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{11}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{1o}]; \\
\vee \\
\vdots \\
\vee \\
\text{IF } [x_1\ \mathsf{V}_{i1}] \vee \ldots \vee [x_m\ \mathsf{V}_{im}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{i1}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{io}]; \\
\vee \\
\vdots \\
\vee \\
\text{IF } [x_1\ \mathsf{V}_{n1}] \vee \ldots \vee [x_m\ \mathsf{V}_{nm}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{n1}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{no}].
\end{array} \tag{12}
$$

$$
\begin{array}{l}
\text{IF } [x_1\ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{1m}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{11}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{1o}]; \\
\wedge \\
\vdots \\
\wedge \\
\text{IF } [x_1\ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{im}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{i1}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{io}]; \\
\wedge \\
\vdots \\
\wedge \\
\text{IF } [x_1\ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m\ \mathsf{V}_{nm}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{n1}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{no}].
\end{array} \tag{13}
$$

$$
\begin{array}{l}
\text{IF } [x_1\ \mathsf{V}_{11}] \vee \ldots \vee [x_m\ \mathsf{V}_{1m}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{11}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{1o}]; \\
\wedge \\
\vdots \\
\wedge \\
\text{IF } [x_1\ \mathsf{V}_{i1}] \vee \ldots \vee [x_m\ \mathsf{V}_{im}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{i1}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{io}]; \\
\wedge \\
\vdots \\
\wedge \\
\text{IF } [x_1\ \mathsf{V}_{n1}] \vee \ldots \vee [x_m\ \mathsf{V}_{nm}], \text{ THEN } [y_1\ \widetilde{\mathsf{V}}_{n1}] \vee \ldots \vee [y_o\ \widetilde{\mathsf{V}}_{no}].
\end{array} \tag{14}
$$

Similarly, we can transform the MIMO verb rule base (11) into the following MISO verb rule base.

$$
\begin{aligned}
&\{\text{IF } [x_1 \ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{11}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{1o}]\} \\
&\vee \\
&\vdots \\
&\vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{i1}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{io}]\} \\
&\vee \\
&\vdots \\
&\vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{n1}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{no}]\}
\end{aligned} \tag{15}
$$

The MIMO verb rule base (12) can be transformed into the following MISO verb rule base.

$$
\begin{aligned}
&\{\text{IF } [x_1 \ \mathsf{V}_{11}] \vee \ldots \vee [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{11}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{11}] \vee \ldots \vee [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{1o}]\} \\
&\vee \\
&\vdots \\
&\vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \vee \ldots \vee [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{i1}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \vee \ldots \vee [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{io}]\} \\
&\vee \\
&\vdots \\
&\vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \vee \ldots \vee [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{n1}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \vee \ldots \vee [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{no}]\}
\end{aligned} \tag{16}
$$

The MIMO verb rule base (13) can be transformed into the following MISO verb rule base.

$$
\begin{aligned}
&\{\text{IF } [x_1 \ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{11}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{11}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{1o}]\} \\
&\wedge \\
&\vdots \\
&\wedge \\
&\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{i1}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{io}]\} \\
&\wedge \\
&\vdots \\
&\wedge \\
&\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{n1}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \wedge \ldots \wedge [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{no}]\}
\end{aligned} \tag{17}
$$

The MIMO verb rule base (14) can be transformed into the following MISO verb rule base.

$$
\begin{aligned}
&\{\text{IF } [x_1 \ \mathsf{V}_{11}] \vee \ldots \vee [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{11}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{11}] \vee \ldots \vee [x_m \ \mathsf{V}_{1m}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{1o}]\} \\
&\wedge \\
&\vdots \\
&\wedge \\
&\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \vee \ldots \vee [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{i1}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{i1}] \vee \ldots \vee [x_m \ \mathsf{V}_{im}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{io}]\} \\
&\wedge \\
&\vdots \\
&\wedge \\
&\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \vee \ldots \vee [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_1 \ \widetilde{\mathsf{V}}_{n1}]\} \\
&\vee \ldots \vee \\
&\{\text{IF } [x_1 \ \mathsf{V}_{n1}] \vee \ldots \vee [x_m \ \mathsf{V}_{nm}], \text{ THEN } [y_o \ \widetilde{\mathsf{V}}_{no}]\}
\end{aligned} \tag{18}
$$

## IV. Properties of Computational Verb Rule Bases

The basic properties of verb rule bases are defined as follows.

*Definition 1 (Complete):* A computational verb rule base is *complete* if and only if all possible permutations of computational verb phrases of the inputs are included in the antecedents of the rule base.

*Definition 2 (Incomplete):* A computational verb rule base is *incomplete* if and only if at least one permutation of computational verb phrases of the inputs is missing in the antecedents of the rule base.

*Definition 3 (Exhaustive):* A computational verb rule base is *exhaustive* if and only if all possible permutations of computational verb phrases of the outputs are included in the consequences of the rule base.

*Definition 4 (Non-exhaustive):* A computational verb rule base is *non-exhaustive* if and only if at least one permutation of computational verb phrases of the outputs is not included in the consequences of the rule base.

*Definition 5 (Consistent):* A computational verb rule base is *consistent* if and only if each permutation of computational verb phrases in the input is mapped onto only one permutation of computational verb phrases of the outputs.

*Definition 6 (inconsistent):* A computational verb rule base is *inconsistent* if and only if at least one permutation of computational verb phrases in the input is mapped onto more than one permutation of computational verb phrases of the outputs.

*Definition 7 (Monotonic):* A computational verb rule base is *monotonic* if and only if each permutation of computational verb phrases in the output is mapped from only one permutation of computational verb phrases of the inputs.

*Definition 8 (Non-monotonic):* A computational verb rule base is *non-monotonic* if and only if at least one permutation of computational verb phrases in the output is mapped from more than one permutation of computational verb phrases of the inputs.

A very good computational verb rule base is most likely complete, non-exhaustive, non-monotonic and must be consistent. However, in real-life applications, a useful computational verb rule base doesn't need to be "very good" in order to be "useful". The degree of usefulness of computational verb rule bases with all permutations of different properties are listed in Table I where the last column shows the degree of usefulness of the rule base. The degree of usefulness are lumped into three categories represented by H(hight), M(medium) and L(low). Observe that the consistent and complete verb rule bases are of high degree of usefulness, the consistent and incomplete ones are of medium degree of usefulness while the inconsistent ones are of low degree of usefulness. Since a verb rule base of M degree of usefulness is incomplete, we can upgrade it into H degree by constructing additional verb rules based on additional observations.

## V. Formal Presentation of Computational Verb Rule Bases Using Boolean Matrices

*Definition 9 (Boolean matrix):* An $m \times n$ *Boolean matrix* is a matrix consisting of $m$ rows and $n$ columns and, of which each element can only take 0 or 1 as its value.

The formal presentation of a computational verb rule base using Boolean matrix is constructed as follows.

1) Label the rows of the Boolean matrix with *all* permutations of computational verb phrases of inputs;
2) Label the columns of the Boolean matrix with *all* permutations of computational verb phrases of outputs;
3) If an element corresponds to a mapping between an input permutation to an output permutation, then set its value to 1. Otherwise, set its value to 0.

TABLE I

The degree of usefulness of computational verb rule bases with permutations of different properties of computational verb rule bases (Y=yes and N=no).

| no. | complete | exhaustive | consistent | monotonic | degree |
|---|---|---|---|---|---|
| 1 | Y | Y | Y | Y | H |
| 2 | Y | Y | Y | N | H |
| 3 | Y | N | Y | Y | H |
| 4 | Y | N | Y | N | H |
| 5 | N | Y | Y | Y | M |
| 6 | N | Y | Y | N | M |
| 7 | N | N | Y | Y | M |
| 8 | N | N | Y | N | M |
| 9 | Y | Y | N | Y | L |
| 10 | Y | Y | N | N | L |
| 11 | Y | N | N | Y | L |
| 12 | Y | N | N | N | L |
| 13 | N | Y | N | Y | L |
| 14 | N | Y | N | N | L |
| 15 | N | N | N | Y | L |
| 16 | N | N | N | N | L |

TABLE II

Labeling the Boolean matrix for verb rule base (19)

| Input/output | II | ID | DI | DD |
|---|---|---|---|---|
| III | 0 | 0 | 0 | 1 |
| IID | 0 | 0 | 0 | 0 |
| IDI | 0 | 0 | 0 | 0 |
| IDD | 0 | 0 | 0 | 0 |
| DII | 0 | 1 | 0 | 0 |
| DID | 0 | 0 | 0 | 0 |
| DDI | 1 | 0 | 0 | 0 |
| DDD | 0 | 0 | 0 | 0 |

*Example 1:* Consider the following verb rule base of which both input and output take two computational verbs increase and decrease.

$$\begin{aligned}
&\text{IF } [x_1 \text{ increase}] \wedge [x_2 \text{ increase}] \wedge [x_3 \text{ increase}],\\
&\text{THEN } [y_1 \text{ decrease}] \wedge [y_2 \text{ decrease}];\\
&\text{IF } [x_1 \text{ decrease}] \wedge [x_2 \text{ increase}] \wedge [x_3 \text{ increase}],\\
&\text{THEN } [y_1 \text{ increase}] \wedge [y_2 \text{ decrease}];\\
&\text{IF } [x_1 \text{ decrease}] \wedge [x_2 \text{ decrease}] \wedge [x_3 \text{ increase}],\\
&\text{THEN } [y_1 \text{ increase}] \wedge [y_2 \text{ increase}].
\end{aligned} \tag{19}$$

The Boolean matrix is constructed as shown in Table II and given by

$$\begin{pmatrix}
0 & 0 & 0 & 1\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 1 & 0 & 0\\
0 & 0 & 0 & 0\\
1 & 0 & 0 & 0\\
0 & 0 & 0 & 0
\end{pmatrix}. \tag{20}$$

Based on the formal presentation of verb rule bases using Boolean matrices, the properties of verb rule bases can be easily defined as follows.

*Definition 10 (Complete):* A verb rule base is *complete* if and only if there is at least one 1 in each row of its Boolean matrix.

*Definition 11 (Incomplete):* A verb rule base is *incomplete* if and only if there is at least one row of its Boolean matrix consists of all 0's.

*Definition 12 (Exhaustive):* A verb rule base is *exhaustive* if and only if there is at least one 1 in each column of its Boolean matrix.

*Definition 13 (Non-exhaustive):* A verb rule base is *non-exhaustive* if and only if there is at least one column of its Boolean matrix consists of all 0's.

*Definition 14 (Consistent):* A verb rule base is *consistent* if and only if there is no more than one 1 in each row of its Boolean matrix.

*Definition 15 (inconsistent):* A verb rule base is *inconsistent* if and only if there is at least one row of its Boolean matrix consists of more than one 1.

*Definition 16 (Monotonic):* A verb rule base is *monotonic* if and only if there is no more than one 1 in each column of its Boolean matrix.

*Definition 17 (Non-monotonic):* A verb rule base is *non-monotonic* if and only if there is at least one column of its Boolean matrix consists of more than one 1.

## VI. Merging Computational Verb Rule Bases

Given the following two computational verb rule bases

$$\begin{aligned}
&\mathbb{VB}_1: \\
&\quad \text{IF } [x_1 \mathsf{V}_{i1}] \wedge \ldots \wedge [x_m \mathsf{V}_{im}], \\
&\quad \text{THEN } [y_1 \hat{\mathsf{V}}_{i1}] \wedge \ldots \wedge [y_o \hat{\mathsf{V}}_{io}]; \\
&\mathbb{VB}_2: \\
&\quad \text{IF } [\tilde{x}_1 \tilde{\mathsf{V}}_{j1}] \wedge \ldots \wedge [\tilde{x}_k \tilde{\mathsf{V}}_{jk}], \\
&\quad \text{THEN } [\tilde{y}_1 \bar{\mathsf{V}}_{j1}] \wedge \ldots \wedge [\tilde{y}_p \bar{\mathsf{V}}_{jp}]; \\
&\quad i = 1, \ldots, n; j = 1, \ldots, r.
\end{aligned} \tag{21}$$

Let $\mathfrak{B}_1$ and $\mathfrak{B}_2$ be the Boolean matrices of $\mathbb{VB}_1$ and $\mathbb{VB}_2$, respectively. And assume that $\mathfrak{B}_1$ and $\mathfrak{B}_2$ are $p_1 \times q_1$ and $p_2 \times q_2$ Boolean matrices, respectively. The permutations of verb phrases for inputs used by $\mathbb{VB}_1$ and $\mathbb{VB}_2$ are packed into two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, respectively. $\boldsymbol{x}_1$ is $p_1$ dimensional while $\boldsymbol{x}_2$ is $p_2$ dimensional. The permutations of verb phrases for outputs used by $\mathbb{VB}_1$ and $\mathbb{VB}_2$ are packed into two vectors $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, respectively. $\boldsymbol{y}_1$ is $q_1$ dimensional while $\boldsymbol{y}_2$ is $q_2$ dimensional. Observe that $\boldsymbol{x}_1$ and $\boldsymbol{y}_1$ are row label vector and column label vector of $\mathfrak{B}_1$, respectively. And $\boldsymbol{x}_2$ and $\boldsymbol{y}_2$ are row label vector and column label vector of $\mathfrak{B}_2$, respectively.

Let $\mathfrak{B}$ be the Boolean matrix of the merging verb rule base $\mathbb{VB} = \mathbb{VB}_1 + \mathbb{VB}_2$. $\mathfrak{B}$ is a $p_1p_2 \times q_1q_2$ matrix. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the row label and column label vectors of $\mathfrak{B}$, respectively. $\boldsymbol{x}$ and $\boldsymbol{y}$ are $p_1p_2$ and $q_1q_2$ dimensional matrices, respectively. Then we have

$$\begin{aligned}
\boldsymbol{x} &= \mathrm{rw}(\boldsymbol{x}_1 \odot \boldsymbol{x}_2^\top), \\
\boldsymbol{y} &= \mathrm{rw}(\boldsymbol{y}_1 \odot \boldsymbol{y}_2^\top)
\end{aligned} \tag{22}$$

where $\mathrm{rw}(A)$ denotes the operation of repack elements of matrix $A$ into a vector. For example,

$$\mathrm{rw}\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix}. \tag{23}$$

The symbol "$\odot$" denotes a symbolic dot product illustrated as follow.

$$\begin{aligned}
\begin{pmatrix} 1 \\ 2 \end{pmatrix} \odot \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} &= \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \end{pmatrix}, \\
\begin{pmatrix} a \\ b \end{pmatrix} \odot \begin{pmatrix} a & b & c \end{pmatrix} &= \begin{pmatrix} aa & ab & ac \\ ba & bb & bc \end{pmatrix}, \\
\begin{pmatrix} a \\ b \end{pmatrix} \odot \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} &= \begin{pmatrix} a1 & a2 & a3 \\ b1 & b2 & b3 \end{pmatrix}.
\end{aligned} \tag{24}$$

Let's pack all elements of $\mathfrak{B}_1$ and $\mathfrak{B}_2$ row wise into $p_1q_1$-vector $\boldsymbol{v}_1$ and $p_2q_2$-vector $\boldsymbol{v}_2$, respectively. Calculate a $p_1q_1 \times p_2q_2$ matrix $A$ as

$$A = \boldsymbol{v}_1\boldsymbol{v}_2^\top = \mathrm{rw}(\mathfrak{B}_1)[\mathrm{rw}(\mathfrak{B}_2)]^\top. \tag{25}$$

We split $A$ into a $p_1 \times p_2$ block matrix as follow

$$A = \begin{pmatrix} A_{11} & \ldots & A_{1p_2} \\ \vdots & \ddots & \vdots \\ A_{p_11} & \ldots & A_{p_1p_2} \end{pmatrix} \tag{26}$$

where $A_{ij}, i = 1, \ldots, p_1; j = 1, \ldots, p_2$ are $q_1 \times q_2$ matrix. Then we first repack all $A_{ij}$ row-wisely and then repack each $A_{ij}$ row-wisely to form the Boolean matrix as

$$\mathfrak{B} = \begin{pmatrix} \{\mathrm{rw}(A_{11})\}^\top \\ \vdots \\ \{\mathrm{rw}(A_{1p_2})\}^\top \\ \vdots \\ \{\mathrm{rw}(A_{p_11})\}^\top \\ \vdots \\ \{\mathrm{rw}(A_{p_1p_2})\}^\top \end{pmatrix} \tag{27}$$

*Example 2:* Let $\mathfrak{B}_1$ and $\mathfrak{B}_2$ be

$$\mathfrak{B}_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \mathfrak{B}_2 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \tag{28}$$

the row label and column label vectors of $\mathfrak{B}_1$ are given by

$$\boldsymbol{x}_1 = \begin{pmatrix} a \\ b \end{pmatrix}, \boldsymbol{y}_1 = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \tag{29}$$

and row label and column label vectors of $\mathfrak{B}_2$ are given by

$$\boldsymbol{x}_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \boldsymbol{y}_2 = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix}. \tag{30}$$

The row label vector and column label vector of $\mathfrak{B}$ are given by

$$\boldsymbol{x} = \begin{pmatrix} a1 \\ a2 \\ b1 \\ b2 \end{pmatrix}, \boldsymbol{y} = \begin{pmatrix} \alpha 3 \\ \alpha 4 \\ \alpha 5 \\ \beta 3 \\ \beta 4 \\ \beta 5 \end{pmatrix}. \tag{31}$$

We calculate matrix $A$ as

$$\begin{aligned} A &= \mathrm{rw}(\mathfrak{B}_1)[\mathrm{rw}(\mathfrak{B}_2)]^\top \\ &= \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}^\top \\ &= \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}, \end{aligned} \tag{32}$$

from which the block matrices are given by

$$\begin{aligned} A_{11} &= \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, A_{12} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \\ A_{21} &= \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, A_{22} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}. \end{aligned} \tag{33}$$

Then $\mathfrak{B}$ is given by

$$\mathfrak{B} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}. \tag{34}$$

A fast algorithm to calculate $\mathfrak{B}$ is given as follow.

*Algorithm 1:* Let $\mathfrak{B}_1 = \{b_1(i,j)\}$, then $\mathfrak{B}$ can be constructed as

$$\mathfrak{B} = \begin{pmatrix} b_1(1,1)\mathfrak{B}_2 & \dots & b_1(1,q_1)\mathfrak{B}_2 \\ \vdots & \ddots & \vdots \\ b_1(p_1,1)\mathfrak{B}_2 & \dots & b_1(p_1,q_1)\mathfrak{B}_2 \end{pmatrix}. \tag{35}$$

*Example 3:* Let us use the fast algorithm to redo Example 2.

$$\begin{aligned} \mathfrak{B} &= \begin{pmatrix} 1\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} & 0\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \\ 1\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} & 1\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}. \end{aligned} \tag{36}$$

## VII. Splitting Computational Verb Rule Bases

The reverse process of merging verb rule bases as presented in Section VI is to split a verb rule base into two verb rule bases. It follows from Algorithm 1 that only $\mathbb{VB}$ with Boolean matrix $\mathfrak{B}$ of the form in Eq. (35) can be split into two $\mathbb{VB}$'s.

*Algorithm 2:* For any $p_1p_2 \times q_1q_2$ Boolean matrix $\mathfrak{B}$, let us split it into $p_1 \times q_1$ block matrices $\{A_{ij}\}$. If $A_{ij}$'s can be categorized two classes, one of which consists of all zero elements and the other of which consists of at least one 1, then $\mathfrak{B}$ can be split into the following two Boolean matrices.

$$\begin{aligned} &\mathfrak{B}_1 = \{b_{ij}\}, b_{ij} = \begin{cases} 0, & \text{if } A_{ij} \text{ contains no 1,} \\ 1, & \text{if } A_{ij} \text{ contains at least one 1,} \end{cases} \\ &\mathfrak{B}_2 = A_{ij}, \text{ if } A_{ij} \text{ contains at least one 1.} \end{aligned} \tag{37}$$

*Example 4:*

$$\mathfrak{B} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{38}$$

Observe that if we choose $A_{ij}$ to be $2 \times 3$ matrices, then $A_{ij}$ can be categorized into the following two types:

$$A_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}, A_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{39}$$

And we have

$$\mathfrak{B} = \begin{pmatrix} A_1 & A_0 & A_1 \\ A_0 & A_1 & A_0 \end{pmatrix} \tag{40}$$

Therefore, $\mathfrak{B}$ can be split into

$$\mathfrak{B}_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \mathfrak{B}_2 = A_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}. \tag{41}$$

*Example 5:* In Example 4 if we choose

$$A_1 = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}, A_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}, \tag{42}$$

then

$$\mathfrak{B} = \begin{pmatrix} A_1 & A_0 & A_1 \\ A_1 & A_0 & A_1 \\ A_0 & A_1 & A_0 \\ A_0 & A_1 & A_0 \end{pmatrix}. \tag{43}$$

Therefore, $\mathfrak{B}$ can be split into

$$\mathfrak{B}_1 = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \mathfrak{B}_2 = A_1 = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}. \tag{44}$$

Observe from Examples 4 and 5 that some verb rule bases can be split into different sets of verb rule bases.

## VIII. Concluding Remarks

To manage the complexity of verb rule models of engineering systems, we need to either merge or split verb rule bases. When two verb rule bases are closely related to each other, we usually merge them into one. On the other hand, if it becomes too difficult to handel a big verb rule set, we usually split it into smaller verb rule bases. However, not all big rule bases can be split into smaller ones. Boolean matrices provide us with an efficient tool to study the properties of verb rule bases and a tool to merge and split verb rule bases. However, to split a big verb rule base into smaller ones is lack of general method though Algorithm 2 provides an easy way to split verb rule bases of special configurations. To split big verb rule bases is much more important than to merge small verb rule bases because in engineering we usually divide a complex problem into controllable small ones. Therefore, future studies will be focused on splitting verb rules bases.

## REFERENCES


[1] Guanrong Chen and Trung Tat Pham. *Introduction to Fuzzy Systems*. Chapman & Hall/CRC, November 2005. ISBN:1-58488-531-9.

[2] Wanmi Chen, Yanqin Wei, Minrui Fei, and Huosheng Hu. Applications of computational verbs to image processing of RoboCup small-size robots. In *Intelligent Control and Automation*, volume 344/2006 of *Lecture Notes in Control and Information Sciences*, pages 494–499. Springer, Berlin / Heidelberg, 2006.

[3] Yi Guo. A study of adverbs as modifiers of computational verbs. *International Journal of Computational Cognition*, 6(1):31–35, March 2008 [available online at $http://www.YangSky.us/ijcc/ijcc61.htm$, $http://www.YangSky.com/ijcc/ijcc61.htm$].

[4] Yang's Scientific Research Institute LLC. **FireEye** *Visual Flame Detecting Systems*. http://www.yangsky.us/products/flamesky/index.htm, http://www.yangsky.com/products/flamesky/index.htm, 2005.

[5] Yang's Scientific Research Institute LLC. **BarSeer** *Webcam Barcode Scanner*. http://www.yangsky.us/demos/barseer/barseer.htm, http://www.yangsky.com/demos/barseer/barseer.htm, 2006.

[6] Yang's Scientific Research Institute LLC. *Cognitive Stock Charts*. http://www.yangsky.us/products/stock/, http://www.yangsky.com/products/stock/, 2006.

[7] Yang's Scientific Research Institute LLC. **PornSeer** *Pornographic Image and Video Detection Systems*. http://www.yangsky.us/products/dshowseer/porndetection/PornSeePro.htm, http://www.yangsky.com/products/dshowseer/porndetection/PornSeePro.htm, 2006.

[8] Yang's Scientific Research Institute LLC. and Wuxi Xingcard Technology Ltd. **YangSky-MAGIC** *Visual Card Counters*. http://www.yangsky.us/products/cardsky/cardsky.htm, http://www.yangsky.com/products/cardsky/cardsky.htm, 2004.

[9] Yang's Scientific Research Institute LLC. and Chinese Traffic Management Research Institute of the Ministry of Public Security(TMRI-China). **DriveQfy** *Automatic CCTV Driver Qualify Testing Systems*. http://www.yangsky.us/products/driveqfy/driveqfy.htm, http://www.yangsky.com/products/driveqfy/driveqfy.htm, 2005.

[10] R. Tonelli and T. Yang. Synchronizing Hénon maps using computational verb controllers. *Phys. Rev. E.*, 2007. submitted.

[11] R. Tonelli and T. Yang. Controlling Chua's circuits using computational verb controllers. *International Journal of Robust and Nonlinear Control*, Apr. 17 2008.

[12] H.-B. Wang and T. Yang. Training neural networks using computational verb rules. *International Journal of Computational Cognition*, 6(2):17–32, June 2008 [available online at $http://www.YangSky.us/ijcc/ijcc62.htm$, $http://www.YangSky.com/ijcc/ijcc62.htm$].

[13] T. Yang. Verbal paradigms—Part I: Modeling with verbs. Technical Report Memorandum No. UCB/ERL M97/64, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, 9 Sept. 1997. page 1-15.

[14] T. Yang. Verbal paradigms—Part II: Computing with verbs. Technical Report Memorandum No. UCB/ERL M97/66, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, 18 Sept. 1997. page 1-27.

[15] T. Yang. Computational verb systems: Computing with verbs and applications. *International Journal of General Systems*, 28(1):1–36, 1999.

[16] T. Yang. Computational verb systems: Adverbs and adverbials as modifiers of verbs. *Information Sciences*, 121(1-2):39–60, Dec. 1999.

[17] T. Yang. Computational verb systems: Modeling with verbs and applications. *Information Sciences*, 117(3-4):147–175, Aug. 1999.

[18] T. Yang. Computational verb systems: Verb logic. *International Journal of Intelligent Systems*, 14(11):1071–1087, Nov. 1999.

[19] T. Yang. Computational verb systems: A new paradigm for artificial intelligence. *Information Sciences—An International Journal*, 124(1-4):103–123, 2000.

[20] T. Yang. Computational verb systems: Verb predictions and their applications. *International Journal of Intelligent Systems*, 15(11):1087–1102, Nov. 2000.

[21] T. Yang. Computational verb systems: Verb sets. *International Journal of General Systems*, 20(6):941–964, 2000.

[22] T. Yang. *Advances in Computational Verb Systems*. Nova Science Publishers, Inc., Huntington, NY, May 2001. ISBN 1-56072-971-6.

[23] T. Yang. Computational verb systems: Computing with perceptions of dynamics. *Information Sciences*, 134(1-4):167–248, Jun. 2001.

[24] T. Yang. Computational verb systems: The paradox of the liar. *International Journal of Intelligent Systems*, 16(9):1053–1067, Sept. 2001.

[25] T. Yang. Computational verb systems: Verb numbers. *International Journal of Intelligent Systems*, 16(5):655–678, May 2001.

[26] T. Yang. *Impulsive Control Theory*, volume 272 of *Lecture Notes in Control and Information Sciences*. Spinger-Verlag, Berlin, Aug. 2001. ISBN 354042296X.

[27] T. Yang. *Computational Verb Theory: From Engineering, Dynamic Systems to Physical Linguistics*, volume 2 of *YangSky.com Monographs in Information Sciences*. Yang's Scientific Research Institute, Tucson, AZ, Oct. 2002. ISBN:0-9721212-1-8.

[28] T. Yang. Computational verb systems: Verbs and dynamic systems. *International Journal of Computational Cognition*, 1(3):1–50, Sept. 2003.

[29] T. Yang. *Fuzzy Dynamic Systems and Computational Verbs Represented by Fuzzy Mathematics*, volume 3 of *YangSky.com Monographs in Information Sciences*. Yang's Scientific Press, Tucson, AZ, Sept. 2003. ISBN:0-9721212-2-6.

[30] T. Yang. *Physical Linguistics: Measurable Linguistics and Duality Between Universe and Cognition*, volume 5 of *YangSky.com Monographs in Information Sciences*. Yang's Scientific Press, Tucson, AZ, Dec. 2004.

[31] T. Yang. Simulating human cognition using computational verb theory. *Journal of Shanghai University(Natural Sciences)*, 10(s):133–142, Oct. 2004.

[32] T. Yang. Architectures of computational verb controllers: Towards a new paradigm of intelligent control. *International Journal of Computational Cognition*, 3(2):74–101, June 2005 [available online at $http://www.YangSky.com/ijcc/ijcc32.htm$, $http://www.YangSky.us/ijcc/ijcc32.htm$].

[33] T. Yang. Applications of computational verbs to the design of P-controllers. *International Journal of Computational Cognition*, 3(2):52–60, June 2005 [available online at $http://www.YangSky.us/ijcc/ijcc32.htm$, $http://www.YangSky.com/ijcc/ijcc32.htm$].

[34] T. Yang. Applications of computational verbs to digital image processing. *International Journal of Computational Cognition*, 3(3):31–40, September 2005 [available online at $http://www.YangSky.us/ijcc/ijcc33.htm$, $http://www.YangSky.com/ijcc/ijcc33.htm$].

[35] T. Yang. Bridging the Universe and the Cognition. *International Journal of Computational Cognition*, 3(4):1–15, December 2005 [available online at $http://www.YangSky.us/ijcc/ijcc34.htm$, $http://www.YangSky.com/ijcc/ijcc34.htm$].

[36] T. Yang. Applications of computational verbs to effective and realtime image understanding. *International Journal of Computational Cognition*, 4(1):49–67, March 2006 [available online at $http://www.YangSky.com/ijcc/ijcc41.htm$, $http://www.YangSky.us/ijcc/ijcc41.htm$].

[37] T. Yang. Applications of computational verbs to feeling retrieval from texts. *International Journal of Computational Cognition*, 4(3):28–45, September 2006 [available online at $http://www.YangSky.com/ijcc/ijcc43.htm$, $http://www.YangSky.us/ijcc/ijcc43.htm$].

[38] T. Yang. Rule-wise linear computational verb systems: Dynamics and control. *International Journal of Computational Cognition*, 4(4):18–33, December 2006 [available online at $http://www.YangSky.com/ijcc/ijcc44.htm$, $http://www.YangSky.us/ijcc/ijcc44.htm$].

[39] T. Yang. Applications of computational verbs to cognitive models of stock markets. *International Journal of Computational Cognition*, 4(2):1–13, June 2006 [available online at $http://www.YangSky.us/ijcc/ijcc42.htm$, $http://www.YangSky.com/ijcc/ijcc42.htm$].

[40] T. Yang. Applications of computational verbs to the study of the effects of Russell's annual index reconstitution on stock markets. *International Journal of Computational Cognition*, 4(3):1–8, September 2006 [available online at $http://www.YangSky.us/ijcc/ijcc43.htm$, $http://www.YangSky.com/ijcc/ijcc43.htm$].

[41] T. Yang. Bridging computational verbs and fuzzy membership functions using computational verb collapses. *International Journal of Computational Cognition*, 4(4):47–61, December 2006 [available online at $http://www.YangSky.us/ijcc/ijcc44.htm$, $http://www.YangSky.com/ijcc/ijcc44.htm$].

[42] T. Yang. Computational verb decision trees. *International Journal of Computational Cognition*, 4(4):34–46, December 2006 [available online at $http://www.YangSky.us/ijcc/ijcc44.htm$, $http://www.YangSky.com/ijcc/ijcc44.htm$].

[43] T. Yang. Distances and similarities of saturated computational verbs. *International Journal of Computational Cognition*, 4(4):62–77, December 2006 [available online at $http://www.YangSky.us/ijcc/ijcc44.htm$, $http://www.YangSky.com/ijcc/ijcc44.htm$].

[44] T. Yang. Stable computational verb controllers. *International Journal of Computational Cognition*, 4(4):9–17, December 2006 [available online at $http://www.YangSky.us/ijcc/ijcc44.htm$, $http://www.YangSky.com/ijcc/ijcc44.htm$].

[45] T. Yang. Using computational verbs to cluster trajectories and curves. *International Journal of Computational Cognition*, 4(4):78–87, December 2006 [available online at $http://www.YangSky.us/ijcc/ijcc44.htm$, $http://www.YangSky.com/ijcc/ijcc44.htm$].

[46] T. Yang. Accurate video flame-detecting system based on computational verb theory. AS *Installer*, (42):154–157, August 2007. (in Chinese).

[47] T. Yang. *The Mathematical Principles of Natural Languages: The First Course in Physical Linguistics*, volume 6 of *YangSky.com Monographs in Information Sciences*. Yang's Scientific Press, Tucson, AZ, Dec. 2007. ISBN:0-9721212-4-2.

[48] T. Yang. Applications of computational verb theory to the design of accurate video flame-detecting systems. *International Journal of Computational Cognition*, 5(3):25–42, September 2007 [available online at $http://www.YangSky.us/ijcc/ijcc53.htm$, $http://www.YangSky.com/ijcc/ijcc53.htm$].

[49] T. Yang. Cognitive engineering and cognitive industry. *International Journal of Computational Cognition*, 5(3):1–24, September 2007 [available online at $http://www.YangSky.us/ijcc/ijcc53.htm$, $http://www.YangSky.com/ijcc/ijcc53.htm$].

[50] T. Yang. Computational verb neural networks. *International Journal of Computational Cognition*, 5(3):57–62, September 2007 [available online at $http://www.YangSky.us/ijcc/ijcc53.htm$, $http://www.YangSky.com/ijcc/ijcc53.htm$].

[51] T. Yang. Computational verb theory: Ten years later. *International Journal of Computational Cognition*, 5(3):63–86, September 2007 [available online at $http://www.YangSky.us/ijcc/ijcc53.htm$, $http://www.YangSky.com/ijcc/ijcc53.htm$].

[52] T. Yang. Learning computational verb rules. *International Journal of Computational Cognition*, 5(3):43–56, September 2007 [available online at $http://www.YangSky.us/ijcc/ijcc53.htm$, $http://www.YangSky.com/ijcc/ijcc53.htm$].

[53] T. Yang and Y. Guo. Measures of ambiguity of computational verbs based on computational verb collapses. *International Journal of Computational Cognition*, 5(4):1–12, December 2007 [available online at $http://www.YangSky.us/ijcc/ijcc54.htm$, $http://www.YangSky.com/ijcc/ijcc54.htm$].

[54] Jian Zhang and Minrui Fei. Determination of verb similarity in computational verb theory. *International Journal of Computational Cognition*, 3(3):74–77, September 2005 [available online at $http://www.YangSky.us/ijcc/ijcc33.htm$, $http://www.YangSky.com/ijcc/ijcc33.htm$].

[55] Sheng Zhu, Zhong-Jie Wang, Yong Liu, and Bao-Liang Xia. An improvement of the design of computational verb PID-controllers. *System Simulation Technology*, 2(1):25–30, Jan. 2006. (in Chinese).