Introduction to Ubit Semantic Computing

Shengyuan Wu
Independent researcher, Ubit inventor, retired professor, Shandong University, Jinan, China

Abstract - *Compute science faces two basic and hardly solved problem; the first is semantic ambiguity; the second is the semantics only understandable to human, not understandable to machine. The difficulties stem from unstructured bits. Based on a new theory, called as Ubit theory, this paper introduces a new generation of semantic computing, called as Ubit Semantic Computing (USC), which can solve the problems successfully. This paper first introduces Ubit theory briefly, then explains how to eliminate semantic ambiguities of natural language, video, audio, and image; and how to make them understandable to both machine and human; then introduces how to make semantics of program and web understandable to both machine and human; making semantic translating tools no needed any more, such as compiler, interpreter, semantic analysis, web parser, domain name resolution; then introduces an integrated interface of USC, by which, anyone can access anything, from anywhere, and in anytime; at last presents an architecture of USC.*

Keywords： Ubit; Ucode; semantic; understandable; content; intent

1. INTRODUCTION

J. Glenn Brookshear says: “Computer scientists dream that the source program is not forced translated to machine language; and machine could perform algorithm discovery process rather than just obeyed execution. [1]”

Tim Berners-Lee points out: "The Web was designed as an information space; with the goal that it should be useful not only for human-human communication, but also that machines would be able to participate and help.”

“One of the major obstacles to this has been the fact that most information on the Web is designed for human consumption,” [2]

Therefore, semantic web had to be first expressed in a “machine processable form" ; then transform the machine processable form into machine executable form [2]; which is still not machine understandable form.

The aim of semantic computing is user’s intents can match author’s intents precisely and efficiently.

The semantic computing definition made by IEEE says “Semantic computing concerns with connecting the (often vaguely formulated) intentions of humans with computational content [3].”

Why the dream of Computer scientists can’t become true? Why is semantics represented in ambiguous form? Why the semantics is not represented by machine and human understandable form?

This stems from bit; stems from codes.

Basic on Ubit theory [4], this paper divides bit into Vbit and Ubit; divides code into Vcode and Ucode; and divides semantic computing into Vbit Semantic Computing (VSC) and Ubit Semantic Computing (USC); the basic idea of USC is that the semantics of contents is expressed in the format understandable to both humans and machines.

All existed semantic computing belongs to VSC.

Authors create contents according to author’s intents, users retrieve contents according to user’s intents; intents need containers; contents are intents inside containers; the containers are codes. Contents are the set of all codes.

The intent container of VSC is Vcode; the intent container of USC is Ucode.

Vcode is unstructured; the number of compatible and consistent Vcodes is limited. VSC can’t represent unlimited intents by limited Vcode, Vcode and intent is not mapped one to one; therefore, ambiguities are inevitable.

The length of compatible and consistent Vcode is limited; therefore, no enough room contains machine understandable semantics.

In contrast, Ucode is structured; the number of compatible and consistent Ucodes is unlimited. Unlimited Ucodes can represent unlimited intents.

Ucode and intent is mapped one to one; therefore, ambiguities are eliminated.

The length of compatible and consistent Ucodes is unlimited, and Ucode is structured. Ucode can contain any machine understandable semantics; and also can associate semantics with machine and human. That is Ucode builds a bridge between machine understandable semantics and human understandable semantics.

This paper introduces Ubit theory briefly, then explains how to eliminate semantic ambiguities of natural language, video, audio, and image; and how to make them understandable to both machine and human by experiment examples; then introduces how to make semantics of program and web understandable to both machine and human; then introduces an integrated interface of USC, by the interface, anyone can access anything, from anywhere, and in anytime; at last presents an architecture of USC.

2. A BRIF INTRODUCTION TO UBIT THEORY

2.1 Vbit, Vcode

Bit is the basic unit of information capacity; a bit can have the value of either 1 or 0 only. Since digital computer was born, bit has only acted as value; this kind bit is called as Vbit. Code only consisted of Vbits is called as Vcode. All the existed codes are Vcodes.

Because Vcodes are not structured, Vcodes in a sequence usually distinguished by code length, different length of Vcodes can hardly be distinguished in one Vcode sequence; generally, only Vcodes in one code set can be mixed in a Vcode sequence.

2.2 Ubit, Ucode

Ubit is not acted as value; Ubit is used to represent data structure, making data structured and distinguishable from the most bottoms. [4]

Ubit can be used in dividing a bit sequence into bit groups.

Fig. 1 (a) contains a bit sequence, which is divided into 5 bit groups: 10, 110, 0, 0, 11110.

The bit group above is characterized as:

If a bit group is one bit long, the bit must be 0, for example the third group “0” and fourth group “0”.

If a bit group is more than one bit long, the right most bit must be 0, the others must be 1; for example: 10, 110, 1110.

The left bit neighbor of a group can’t be 1, but the right bit neighbor of a group can be 1 or 0,

There is one and only one bit 0 in one bit group.

A bit sequence can be distinguished into groups by one Ubit rule; there are ten Ubit rules; the most useful is Right 0 Ubit rule.

2.3 Right 0 Ubit rule

Right 0 Ubit rule: Scan a binary 1 and 0 sequence from left to right, if the first bit is 1, continue scanning until a bit 0 is met, then from the first bit 1 to the bit 0 is distinguished as a group; if the first bit is 0, then the bit 0 is distinguished as a group; continue scanning and distinguishing until the end of the sequence, if the last bit of the sequence is not 0, then the last group is an incomplete group.

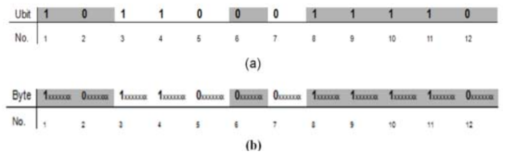

Ubit 1 0 1 1 0 0 0 1 1 1 1 0
No. 1 2 3 4 5 6 7 8 9 10 11 12
(a)
Byte 1xxxxxxx 0xxxxxxx 1xxxxxxx 1xxxxxxx 0xxxxxxx 0xxxxxxx 0xxxxxxx 1xxxxxxx 1xxxxxxx 1xxxxxxx 1xxxxxxx 0xxxxxxx
No. 1 2 3 4 5 6 7 8 9 10 11 12
(b)

Figure 1. *Grouping by Right 0 Ubit rule (a) distinguishing bit sequence into bit groups (b) distinguishing byte sequence into byte groups*

Example 1, distinguish the bit sequence in Fig. 1 (a) into bit groups.

Scan the binary 1 and 0 sequence in Fig 1 (a) from left to right,

The 1st bit is 1, continue scanning, the 2nd bit is 0, therefore, “10” is distinguished as a group;

Continue scanning, the 3rd bit is 1, continue scanning, the 4th bit is 1, continue scanning, the 5th bit is 0, therefore, “110” is distinguished as a group;

Continue scanning, the 6th bit is 0; then, the bit 0 is distinguished as a group.

Continue scanning until the end of the bit sequence; 5 groups are divided by Right 0 Ubit rule.

Here, bit 0 or 1 is used in distinguishing bits into groups, not acted as value. This kind of bit is called as Ubit.

A complete bit group in Figure 1(a), consisting of only Ubits; it is an Ubit group. An Ubit group is called as Uframe.

In Fig. 1(b), each Ubit in (a) is copied to the leftmost bit of each byte respectively; then, the byte sequence is divided into 5 byte groups by the Right 0 Ubit rule, or by Uframe. The distinguishing procedure is the same as example 1; the difference is each Ubit in the leftmost bit of each byte; therefore, the distinguishing is by scanning each Ubit in the leftmost bit of each byte.

Each of the 5 byte groups consists of Ubit and Vbit, each byte group is called as Ucode; the leftmost bit of each byte is Ubit. The length of Ucode in the Ucode sequence is different.

Example 2， distinguish byte sequence into Ucode,

Fig. 2 (a) is a byte sequence, each byte is represented in two hex numbers, the leftmost bit of each byte is Ubit; if the first hex number of a byte is greater than 7, then the Ubit of the byte is 1, else the Ubit of the byte is 0; (here, the old display system takes Ubit as value) for example, the first underline three bytes are displayed as {F2 A0 2D}, the Ubits of three bytes are {1,1,0}, respectively.

The Ubits of the 16 bytes of the first row are 1101101101101110

Then, the 16 byte sequence is distinguished into 4 Ucode by Right 0 Ubit rule.

The underline Ucodes in Fig. 2 (a) are 3, 3, 3, 4, 2, 6 bytes and 1 byte long, respectively. The Ubits of the underline Ucodes are {1,1,0},{1,1,0},{1,1,0}, {1,1,1,0}, {1,0}, {1,1,1, 1,1, 0}, { 0 }, respectively.

It is clear that variable length Ucodes are mixed in a code sequence, compatible with one another.

An Ucode consists of sections, here, the section is byte; the section can be any long, but the shortest section is 2 bits.

Here, Ubit is assigned in the leftmost bit; it can be positioned in any bit of the section as needed.

A section sequence can be distinguished into Ucodes by Ubit rule.

Thus, bit is divided into Vbit and Ubit.

Vbit is used to represent data value; Ubit is used to represent data structure, making data structured and distinguishable from the most bottoms.

Code is divided into Vcode and Ucode.

Vcode only consists of Vbit, unstructured and undistinguishable. All existing codes belong to Vcodes.

Ucode consists of Vbit and Ubit; the Ubits in an Ucode represent the structure of the Ucode, making it structured and distinguishable; the Vbits in an Ucode represent the value of the

Ucode. Variable length Ucodes are compatible with one another; unlimited amount codes are available; the length of Ucode can be very short, or very long, it can contain any kind of semantics, and any amount of semantics; it can also associate with semantics; suitable as contents containers; Ucode acts as an bridge between human and machine understandable semantics.

Ubit theory is a fundamental theory of computer science, it studies how represent data structured from the bit level, how make semantics of contents expressed in the format understandable to both human and machine; in another hand, Ubit theory also studies how to make semantics of private contents not understandable to unauthentic persons and machines.

3 HUMAN AND MACHINE UNDERSTANDABLE SEMANTICS

3.1 Outline of human and machine understandable semantics

Fig. 2 (a) is an Ucode sequence understandable to machine, containing 1, 2, 3, 4, or 6 byte length of Ucodes.

Fig. 2 (b) is a notation group sequence, the human understandable semantics. Each notation group is an object, each object is associated with one and only one Ucode in Fig. 2 (a), The underlined notation groups are audio object, video object, image object, two-Chinese character word object, one-Chinese character word object, three-Chinese character word object, and one English letter object respectively.

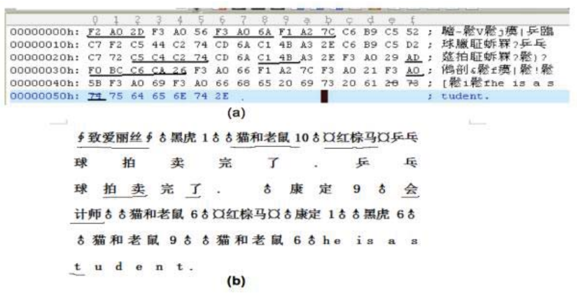
3.2 Human and machine understandable semantics of audio object, video object and image object

3.2.1 Human understandable semantics of audio, video or image object associated by Ucode

In Fig. 2 (b), the symbol pair of "∮∮", "♁♁" and "¤¤" represents audio, video and image object respectively, the character string inside each symbol pair of "∮∮", "♁♁" and "¤¤" is the name of the object. These notations make the objects understandable to human. For example, the first object is an audio object: ∮致爱丽丝∮, the second is video object: ♁黑虎 1♁, the third object is video object: ♁猫和老鼠 10♁(video clip of cat 10) , and the fourth is an image object: ¤红棕马¤. As shown in Table 1; the notation of each video clip of cat and mouse is associated by its related Ucode.

3.2.2 Machine understandable what kind of object semantics embedded in Ucode

In Fig. 2 (a), each audio, video or image object is coded in a 3 byte Ucode; the first bit of each byte is Ubit, the others are Vbit. Machine can distinguish each Ucode by Ubit rules [4]. The type semantics of the audio, video and image object is contained in the first byte of the related Ucode, that is the Ucode is divided into two parts, the first byte is for object type; the Ucode is structured. For example, the first underlined Ucode {F2 A0 2D} is an audio object, the second underlined Ucode {F3 A0 6A} is a video object; the third underlined Ucode {F1 A2 7C} is an image object; the first byte of the Ucode of audio, video and image object is F1, F2, F3, respectively. Because the leftmost bit is an Ubit, the real value of the first byte is 0x71, 0x72 and 0x73; that means the real type attribution of audio, video and image is 0x71, 0x72 and 0x73.



```
             0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: F2 A0 2D F3 A0 56 F3 A0 6A F1 A2 7C C6 B9 C5 52 ; 瞻-鬆V鬆j獏|乒臨
00000010h: C7 F2 C5 44 C2 74 CD 6A C1 4B A3 2E C6 B9 C5 D2 ; 球臓眐蛴羼?乒乓
00000020h: C7 72 C5 C4 C2 74 CD 6A C1 4B A3 2E F3 A0 29 AD ; 筵拍眐蛴羼?鬆)?
00000030h: F0 BC C6 CA 26 F3 A0 66 F1 A2 7C F3 A0 21 F3 A0 ; 偽剖&鬆f獏|鬆!鬆
00000040h: 5B F3 A0 69 F3 A0 66 68 65 20 69 73 20 61 20 73 ; [鬆i鬆fhe is a s
00000050h: 74 75 64 65 6E 74 2E                            ; tudent.
```

(a)

∮致爱丽丝∮♁黑虎 1♁♁猫和老鼠 10♁¤红棕马¤乒乓球　拍　卖　完　了　．　乒　乓球　拍　卖　完　了　．　♁　康　定　9　♁　会计师♁♁猫和老鼠 6♁¤红棕马¤♁康定 1♁♁黑虎 6♁♁猫和老鼠 9♁♁猫和老鼠 6♁he is a s t u d e n t .

(b)

Figure 2. *Human and machine understandable semantic with mixed objects: audio object, video object, image object, one-Chinese character word object, two-Chinese character word object, three-Chinese character word object, and English letter object. (a) Ucode sequence; (b) Notation group sequence*

The maximum value of the first byte of the Ucode is 127; that means it can hold 127 types of objects for this three byte Ucode.

The ASCHII codes in Fig. 2 are no change; they can be distinguished by Right 0 Ubit rule; therefore, ASCHII code is Ucode. It is easy to make Vcode be Ucode by adjusting a bit here; for example, the Chinese GB2312 in Fig. 2; only the leftmost bit of the last byte of each word is changed from 1 to 0.

It is easy to judge what is Ucode and Vcode, if the codes in a sequence are distinguished by Ubit rule; then, they are Ucodes.

Ucode is hierarchically structured; however, Vcode is not structured. For example:

GB2312 is 2 byte code set in Chinese; JISX 0208 is 2 byte code set in Japan; and KSC 5601 is 2 byte code set in Korea. These are local code sets, unstructured, conflict one another; therefore, they can't be distinguished in a mixed code string.

Unicode is universal or global code set, the relation between global code set and local code sets should be in hierarchical form; however, it's not. Unicode is only a recoded code set; UTF16 contains 65,536 codes, including most of the characters needed for writing system in the world. But, different kinds of codes are mixed in Unicode, not structured.

The codes of GB2312, JISX 0208, KSC 5601 and Unicode can't be put into one singular string, because no way to distinguish what kind of codes.

However, GB2312, JISX 0208, KSC 5601 or Unicode can be easily represented by Ucode, for example, here double byte Ucode is used for GB2312 as default; JISX 0208 and KSC 5601 can be represented by two type of object of the Ucode in Fig. 2(a); Unicode can be divided into 4 classes, each can be represented by one type of object of the Ucode. Then all code sets are hierarchically structured, each set can be distinguished by its type attribution [5]. Local code sets, various code sets in various locations are compatible one another, this is called as space consistent.

Here, the object type attribution is embedded in Ucode, not represented in notation form, understandable to machine.

3.2.3 Machine understandable object addressing semantics associated by Ucode

In order to access an object, machine must need to know the address in storage space. There are two ways: one is embedded the physical address in Ucode, another is associating the address by Ucode.

Usually, an object address is represented by directory path as shown in table 1; this is a notation form, only understandable to human. Therefore, the notation form address must be translated to machine.

Table 1 (a) is Ucode sequence for clips of video “cat and mouse”; (b) Ucode associated with human’s language for clips of video cat and mouse: the notation form; (c) Ucode associated with physical address for clips of video “cat and mouse”.

TABLE I. THE ASSOCIATION BETWEEN UCODE AND SEMANTICS

Ucode	F3A0601	…	F3A069	F3A06A

(a) Ucode sequence for clips of video cat and mouse

Notation	♂猫和老鼠 1♂	…	♂猫和老鼠 9♂	♂猫和老鼠 10♂

(b)Ucode associated with human’s language for clips

File l address		…		

(c) Ucode associated physical address for clips of video cat and mouse

3.3 Human and machine understandable semantics of natural language

The most difficult language might be Chinese; so we take Chinese as an example.

3.3.1 Chinese word segmentation problem

Word segmentation is the problem of dividing a string of written language into its component words. In English, the space is a good approximation of a word divider (word delimiter); however, in Chinese character string, there are no spaces as word delimiter; therefore, word segmentation becomes the most difficult problem in Chinese language processing.

In Fig. 2 (b), the two sentences are segmented different; the meaning is totally different.

The first sentence is segmented by author as: “乒乓” “球拍” “卖” “完” “了” “. ”

The meaning is: Ping pong/ rackets/sell/finished/already.

The second sentence is segmented by author as: “乒乓球” “拍卖” “完” “了” “, ”

The meaning is: Ping pong balls/auction/finished/already.

However, if no word delimiters, it’s very difficult to know the author’s real intents.

The two sentences are displayed the same in Fig. 2 (b) as the following.

“乒乓球拍卖完了. ”

The ambiguity of word segmentation is caused by many to one mapping, multiple characters relate to one word; Ucode eliminates this kind of ambiguity by making each word relates to one and only one Ucode; in Fig. 2 (a), the segmentation semantics has been represented by Ucodes, which can be distinguished by Ubit right 0 rule; therefore, Ucode acts as word delimiters; understandable to machine; then machine can translate the segmentation semantics to human, for example, to display words in different colors.

3.3.2 Chinese polyphone word ambiguous problem

In Fig. 2 (b), the first character of Chinese word “会计师” is a polyphone character with two different phonemes, and the meaning is totally different.

The ambiguity of polyphone word is caused by one to many mapping; one character code relates multiple phonemes and meanings. Ucode eliminates this kind of ambiguity by making multiple Ucodes, each is only related to one phoneme and one meaning. Therefore, the right semantics is distinguished by Ucode as shown in Fig. 2.

However, it can’t do so with Vcode, because no enough compaible Vcodes can be used.

4 HUMAN AND MACHINE UNDERSTANDABLE PROGRAM SEMANTICS

Obviously, programming languages are notations form, only understandable to people, not understandable to machine. Therefore, before a program can be run, it must be translated into machine language by compile or interpreter [6]. However, Ucode source program is totally different.

4.1 Ucode source program

Source program usually includes two parts: instruction part and data part. Instruction part is an Ucode sequence. Take the following program as an example:

yCoordinate = intercept + Slope * xCoordinate

There are four variables in the program; each notation form of them is associated by one Ucode, the human understandable semantics, as shown in Table 2.

TABLE II. UCODE OF VARIABLE ASSOCIATES NOTATION FORM AND ADDRESS

Ucode	Ucode_{11}	Ucode_{12}	Ucode_{13}	Ucode_{14}
Notations	yCoordinate	intercept	Slope	xCoordinate
Address	0x0000	0x0004	0x0008	0x000C

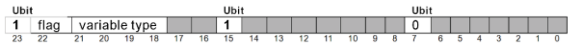
There are three operators in the program; each of them is represented by one Ucode, and the human understandable semantics of each operator is associated by the related Ucode as shown in Table 3.

TABLE III. UCODE OF OPERATOR ASSOCIATES NOTATION FORM

Ucode	Ucode_{21}	Ucode_{22}	Ucode_{23}
Operator	=	+	*

The Ucode format of keyword, operator, and variable is shown in Table 4, in which, 22^{nd} bit is flag, 0 for identifier, 1 for keywords or operators; for variables, the 16 bits, shaded grey, coding the memory address of each variable ; if 22^{nd} flag=1, 21^{st} acted as flag, 0 for keywords, 1 for operators.

TABLE IV. THE UCODE FORMAT OF KEYWORD, OPERATOR, AND VARIABLE



Assume the type of the four variables is float with 4 byte long. Then the relative address is show in Table 2.

The Ucodes of operators are predefined as show in Table 3, and the privilege of the operators is embedded in Ucode by coding the Ucode according to the privilege of operators; for example, here, the value of Ucode$_{23}$ > Ucode$_{22}$ > Ucode$_{21}$. So the operating rules have been embedded in Ucodes.

There is a program developing interface, just like the interface in section VII.

As program inputting, the variable name, such as: yCoordinate, intercept, Slope, xCoordinate, is inputted by user, variable type semantics is also inputted by selecting the type displayed. An Ucode is assigned to each variable, and the related address of each variable is filled in the Ucode according to the memory pointer, which is increased according to the type of variable.

At beginning, the interface initializes a variable table as the Table 2; and initialize the memory pointer=0.

As input "yCoordinate"; the interface first check if the variable has been inside Table 2, if not, then ask user to input the variable type; then assign an Ucode to the variable, here Ucode$_{11}$, and fill the variable type in the Ucode, and put the value of memory pointer into the Ucode as shown in Table 2; then, increase the memory pointer according to the variable type, here increase 4 because the variable type is float, 4 byte long; then put "yCoordinate" into the table 2.

The interface put Ucode$_{11}$ into the source program, and displays "yCoordinate".

Then, input "=", the interface put Ucode$_{21}$ into the program and display "=".

Similarly, input other variable and operators.

At last, the content of the source program is as:

Ucode$_{11}$ Ucode$_{21}$ Ucode$_{12}$ Ucode$_{22}$ Ucode$_{13}$ Ucode$_{23}$ Ucode$_{14}$

And displayed in notation form as:

yCoordinate = intercept + Slope * xCoordinate

This is one statement; the Ucode source file consists of a sequence of Ucode statements, an Ucode can be selected as deliminater among statement; for example, the feedback of ASCHII code.

The notation form can be any character form, not limited to English; and in the source program, user can input remarks, notes by any language.

4.2 The executiion of Ucode source file

The machine knows the data type, the addresses of the variables; the privilege of operators, the operating rules; therefore, the machine knows what instructions should use, what operation sequence, where to access the variables; i.e. how to execute the program.

There are three executable forms: human and machine understandable execution form, suitable for program developing and testing; second, machine understandable execution form; third, machine obeyed execution form.

In human and machine understandable execution form, both human and machine understandable semantics can be displayed.

The machine scanning the Ucode source program as following;

Ucode$_{11}$ Ucode$_{21}$ Ucode$_{12}$ Ucode$_{22}$ Ucode$_{13}$ Ucode$_{23}$ Ucode$_{14}$

From the type attribution of Ucode, machine knows there are three Ucodes of operator: Ucode$_{21}$ Ucode$_{22}$ Ucode$_{23}$ (relate to = , + , * ;); compare the value of Ucode$_{21}$ Ucode$_{22}$ Ucode$_{23}$; (the privilege of the operators by the value of their Ucodes), the highest is: Ucode$_{23}$ (for * *)*; so the machine understands calculating Ucode$_{23}$ first; select an Ucode$_{buffer}$ for storing the middle value; and assign the memory address: x000F. Then, calculate operation "+", then store the value to *Ucode$_{11}$.* Therefore, machine can divide the cource program above into two five Ucode form as the following:

Ucode$_{buffer}$ Ucode$_{21}$Ucode$_{13}$ Ucode$_{23}$ Ucode$_{14}$;

Ucode$_{11}$ Ucode$_{21}$ Ucode$_{12}$ Ucode$_{22}$ Ucode$_{buffer}$.

A complier can generate two three address code after lexical, syntax and semantic analysis [6] as following:

*buffer=Slope * xCoordinate;*

yCoordinate = intercept + buffer.

It is clear that each of five Ucodes form relates to one three address code generated by a complier after lexical, syntax and semantic analysis [6].

Assuming variable: intercept, Slope, xCoordinate have be assigned value: 0.0, 0.5, 2.0, as show in Table 5.

Next, discuss how machine executes the source program in human and machine understandable execution form.

Machine executes the first five Ucodes as following

Ucode$_{buffer}$ Ucode$_{21}$Ucode$_{13}$ Ucode$_{23}$ Ucode$_{14}$;

Display to human as following:

*buffer=Slope * xCoordinate;*

TABLE V. UCODE OF VARIABLE EMBEDDED ADDRESS AND VALUE

Ucode	Ucode$_{11}$	Ucode$_{12}$	Ucode$_{13}$	Ucode$_{14}$	Ucode$_{buffer}$
Address	0x0000	0x0004	0x0008	0x000C	0x000F
Initial Value		0.0	0.5	2.0	
Last value	1.0	0.0	0.5	2.0	1.0

From the type attribution of Ucode, machine knows there are two Ucodes of operator: *Ucode$_{21}$, Ucode$_{23}$* (relate to = , *); compare the value of *Ucode$_{21}$* and *Ucode$_{23}$*; (the privilege of the operators relates the value of their Ucodes), the higher is: *Ucode$_{23}$* ("*"); therefore, First calculate*:*

*Slope * xCoordinate;*

Extract the value in the address embedded in $Ucode_{13}$: address 0x08, value: 0.5; extract the value in the address embedded in $Ucode_{14}$: address 0x000C, value: 2.0;

Do operation of $Ucode_{23}$ "*": 0.5*2.0; and store the result to 0xF;

*buffer= Slope * xCoordinate = 1.0*

The second five Ucodes form is executed in the same manner.

At last, the value in the memory is shown in Table 5.

Human and machine understandable execution form is suitable for program developing and testing;

Machine understandable execution form is similar to human and machine understandable execution form, the difference is that the human understandable notations are not displayed.

The advantages of the human and machine understandable execution from and the machine understandable execution form is as following:

Machine knows the data type, the addresses of the variables; the privilege of operators clearly; therefore, parallelism is much easier. The machine understands what Ucode instructions can be calculated in parallel. The machine can efficiently schedule Ucode instructions, according to the resources available. Therefore, the execution speed is quickened; and the CPU could become the real brain of the machine.

For machine obeyed execution, first extract the machine instruction part and data part according to the machine available resource; and generate an executable file as the existed one.

Because the three execution form does not need compiler; they are machine independent, platform independent and network independent. For machine obeyed execution, the machine is just obeyed to execute an instruction sequence, knows nothing about the program, i.e. not understand the program.

The notations used in Ucode source program can be any language, any notations; there nearly no any syntax requirements. This means you can develop your program by any languages and with great freedom.

5 HUMAN AND MACHINE UNDERSTANDABLE WEB SEMANTICS

The aim of Berners-Lee's Semantic Web is to make web understandable to machine and human.

However, semantic web is only described in notation form, the goal can't be realized. Fortunately, Ucode can achieve the goal easily.

Web tags can be easily represented by Ucodes, for example, each Ucode of one type of three byte Ucodes in Fig. 2 can be used to represented one tag; one type Ucodes can represent about ten thousand tags.

Here, mainly discuss how to represent global address in both human and machine understandable form.

Global address is usually called as Uniform Resource Locator (URL). A typical URL is as the following:

http://example.org/semantic-web/Semantic Web

It's a notation form, only understandable to human; translating to machine is necessary; and the translation is called as domain name resolution.

URL can be divided into two parts, remote address and local address, as shown in Table 6.

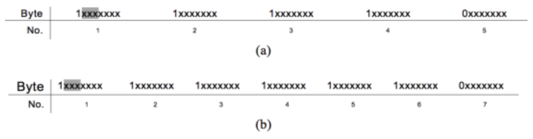
TABLE VI. REMOTE ADDRESS AND LOCAL ADDRESS

$Ucode_{remote}$	$Ucode_{local}$
http://example.org	/ semantic-web/Semantic Web

The local address understandable to machine and human has already discussed as shown in Table 1.

The remote address includes protocol and IP as shown in Table 7.

TABLE VII. DIAGRAM OF REMOTE SEMANTIC ADDRESSING



Byte	1xxxxxxx	1xxxxxxx	1xxxxxxx	1xxxxxxx	0xxxxxxx
No.	1	2	3	4	5

(a)

Byte	1xxxxxxx	1xxxxxxx	1xxxxxxx	1xxxxxxx	1xxxxxxx	1xxxxxxx	0xxxxxxx
No.	1	2	3	4	5	6	7

(b)

Assume the remote address is expressed in 5 byte Ucode format; the first bit of each byte is Ubit, the left 35 bits are Vbit, three Vbits for coding protocol type; the left 32 Vbits for coding IPv 4 as shown in Table 7. (a).

An $Ucode_{Remote}$ in Table 6 is associated with human understandable notation: http://example.org; and the protocol type and the IP address is embedded inside the Ucode. An $Ucode_{Local}$ is similarly as shown in Table 1 (b), which associates with human understandable notation: / semantic-web/Semantic Web, and also associates with the file address associated as shown in Table 1 (c).

Here, "http://example.org/semantic-web/Semantic Web" is described in Ucode form, understandable to machine and human.

The displaying of Ucode URL can be any language, any notations; there nearly no any syntax requirements. This means you can use any character or word to represent your name of URL.

As the $Ucode_{Remote}$ address space is not enough, the length of Ucode can be extended; 7 byte Ucode form as shown in Table 7 (b); then, the address space is 214 times space of IPv 4. Some years later, can be further extended to more bytes, but the old one can continue to use. The current using, the used before, and the shall use in future, are consistent with one another, this is called as time consistent. Therefore, the address space is unlimited and time consistent; domain name resolution no need any more.

6 SECURITY OF USC

Based on Ubit theory, the semantics of public contents can be made easily understandable to human and machine; moreover, the semantics of private contents can also be easily made not understandable to unauthentic persons and machines; therefore, the contents are much safer.

The Ucode contents are structured from the most bottoms, the foundation is secure.

Because machine understandable the languages, no translation needed, the vulnerabilities caused by translation are not existed; for example, DNS attacks are stemmed from domain name resolution; so DNS attacks no exist again.

Software becomes bigger and complicated; vulnerabilities can hardly be avoided in developing stage, updating and modification often needed. Adopting the human and machine understandable execution form, the source program executes step by step; developers can easily debug, easily update software and easily detect and repair vulnerabilities. The inside back door attack is also easily detected.

Computer virus infects a program file by inserting virus code into executed code; which consists of a sequence of instructions and data; because the machine just obeyed to execute the instructions, has no way of knowing what is data and what is program; therefore, virus injection is hardly prevented.

However, with the information inside Ucodes, the machine not only knows what is data and what is program; but also knows the data type, the addresses of the variables; the computation rules clearly. That is the address space is numbered already, variables in program space and data space mapped. It's very difficult for hacker to insert virus into Ucode program.

This is just like the following scenario: when people are waiting on a line, the line is not clearly ordered, not know who is in front and who is behind; then it is easy to cut in. However, if the line is ordered and each one is numbered, each one knows who is in front and who is behind; then it is difficult for someone to cut in line.

Because type and size attribution of each variable is embedded in related Ucode, as source program executing, the machine knows the size of each buffer clearly; overflow attack can be prevented effectively.

The existed security is only based on value operation; the encryption strength only depends on value computing complex [7-9].

USC security is based on both value and structure encryption; the encryption strength not only depends on value computing complex; and also depends on structure computing complex; therefore, much stronger.

USC security can be further strengthened by combining with in-out key, in-out password and in-out nonce. In-out key makes key distribution absolutely secure; in-out password makes password unbreakable; and in-out nonce makes replay attacks useless [10]. Ubit theory combined with in-out key, in-out password and in-out nonce can lay a solid foundation of computer security.

7 AN INTEGRATED INTERFACE OF USC

7.1 Introduction to an experimental interface of USC

An experimental interface of USC is shown in Fig. 3: (a) Interface Menu (only part of the experiment menus listed). (b) Interface, an integrated input method; which is used to create, convert, manage, and retrieve contents by one's intents.

The interface bases on precisely relation between Ucodes and objects, one to one mapped.

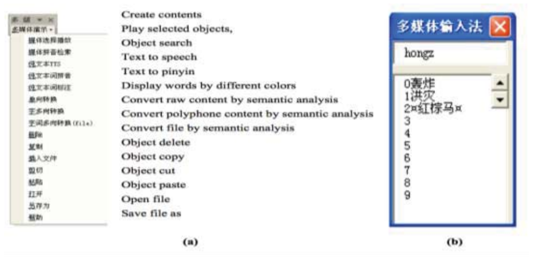


Figure 3. *An experimental Interface of USC (a) Interface Menu; (b) An integrated input method*

(a) Interface Menu

- Creating contents based on author's intents
- Converting raw contents by semantic analysis

"Convert raw contents by semantic analysis", "Convert raw polyphone contents by semantic analysis"

"Convert raw content file by semantic analysis":

The raw content refers to contents with ambiguities.

- Retrieving contents based on user's intents

"Play selected objects", "Object search", "Text to speech", "Text to pinyin", "Display word by different colors"

- Editing menu

"Object delete", "Object copy", "Object cut", "Object paste"

- File menu:

"Open file", "Save file as"

(b) An integrated Interface

The interface is an integrated input method, For example, in (b), input five letters, "hongz", then, three objects are matched in the window: two word objects, each with two characters, and an image object. By this interface, user can interact precisely and semantically with the contents, such as audio, video, image object, Chinese text objects, English text and etc. Anyone from anywhere can access the contents, no matter want kind of languages, want kind of objects.

This interface makes ambiguity no exist; and is machine and human understandable. Author creates contents based on author's intents; user retrieves contents based on user's intents.

7.2 Create semantic contents based on author’s intents

7.2.1 Create video, audio and image object semantic contents based on author’s intents

In Fig. 4 and Fig. 5, three video, an audio and an image objects have been created into the contents.

Each object relates to one, and only one Ucode in the contents; that is author’s intents map to contents one to one.

Each object is distinguishable; the object type embedded in the Ucode; and object address relates to the Ucode. This makes the semantics of contents understandable to machine, machine can access the object directly; no semantic translation needed.

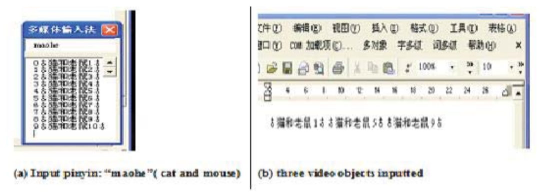

(a) Input pinyin: "maohe"(cat and mouse)
(b) three video objects inputted

Figure 4. *Three video objects inputted*

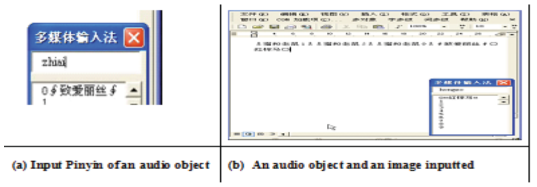

(a) Input Pinyin of an audio object
(b) An audio object and an image inputted

Figure 5. *An audio object and an image object inputted*

The Ucode also relates to human understandable languages. For VSC, each video, audio and image relates to a notation string, not understandable to machine.

7.2.2 Create phoneme semantics based on author’s intents

In Fig. 6, Author inserts a Chinese three character word object with polyphone character: “会计师”into the contents.

The first character is a polyphone character, with two different phoneme, and different meaning; there are two codes, “BBE1” and “ADF0”, and each relates to one phoneme; here “BBE1” is selected according to user’s intent as shown in Fig. 2.

Author’s intents also map to contents one to one.

Further, the three characters can be segmented as: “会计师” or “会计”; here one Ucode maps to “会计师”; it is impossible to be “会计”; so the contents with author’s real intents.

For VSC, the word relates to three characters, one to three mapping, and not one to one mapping. The polyphone character is only expressed by one Vcode: “ADF0”; one Vcode in contents relates to two different intents, neither one to one mapping.

7.2.3 Create word segmentation semantics based on author’s intents

The following example illustrates Chinese word semantics inputting

In Fig. 7, as author inputs “pinpang”, two words: “乒乓”, “乒乓球” appear in the input window; if author

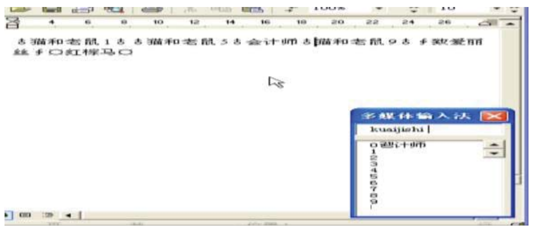


Figure 6. *Insert a Chines three character word object with poly-phoneme character: “会计师” between two video objects: ♁猫和老鼠3 ♁, and ♁猫和老鼠9 ♁.*

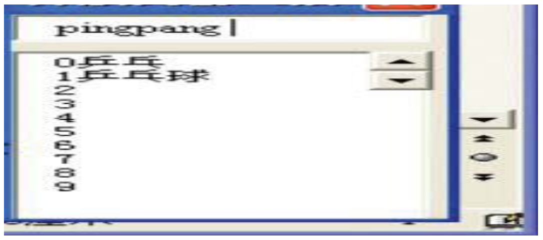


Figure 7. *word segmentation semantics inputting based on author’s intents.*

Selects “乒乓”, then the object “乒乓” is saved as a 4 byte Ucode; if author selects “乒乓球”, then the object “乒乓球” is saved as a 6 byte Ucode as shown in Fig. 2.

Here, Ucode is created by combining the character codes according to user’s selection. Each word is an object, and relates to one and only one Ucode in the contents, author’s intents map contents one to one.

However, for VSC, multiple characters in contents relates to one author’s intents; this is also multiple to one mapping.

7.3 Retrieve semantic contents by user’s intents matching author’s intents

The main characteristics of USC Retrieving are:

Precisely Author’s intents: Author’s intents are expressed precisely;

Precisely User’s intents: User’s intents are expressed precisely.

Precisely matching: User’s intents matching author’s intents precisely;

Directly and efficiently retrieving: because machine understandable semantics, machine can extract the matched contents directly and efficiently.

The content retrieving is based on user’s intent.

As in Fig. 7, if user wants to search object: “乒乓”; “乒乓” is an Ucode, this can only be found in the first sentence of the contents;, if user wants to search object: “乒乓球”; “乒乓球” is an Ucode, this can only be found in the second sentence of the contents; if user wants to search object: “拍卖”, this can only be found in the second sentence. User’s intents map to author’s intents.

User’ retrieving can be in different way: mixed way, it search’s all kinds of contents, all kinds of objects to match user’s intents.

Retrieving in specified area, for example, retrieving from audio objects to match user’s intents, from video object to match user’s intents, from image object to match user’s intents, from one-Chinese character word objects to match user’s intents, from two-Chinese character word objects to match user’s intents; from polyphone words to match user’s intents, from specified languages to match user’s intents, and etc.

In order to be consistent with VSC, retrieving can also be done in VSC retrieving manner.

Display word segmentation semantics to human as retrieving, in Fig. 8, the word semantics is displayed in different colors.

The word segmentation semantics can also be expressed by “Text to speech” as shown in Fig. 9.

There are polyphone word objects in Fig. 10; the word phoneme semantics is distinguished by Pinyin.

乒乓球拍卖完了．乒乓球拍卖完了．

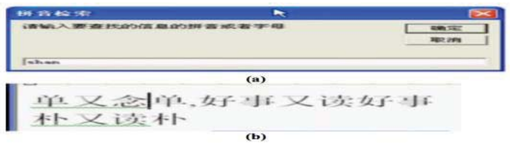
Figure 8. *The word segmentation semantics created by author’s intents is displayed by different colors*

Figure 9. *The word segmentation semantics distinguished by text to speech in different time gaps*

The semantics of polyphone words can be retrieved correctly by user’s intents. There are two “单” in Fig. 10 and 11;

dān yòu niàn shàn hǎo shì yòu dú hǎo shì
单 又 念 单，好 事 又 读 好 事

Figure 10. *displaying the word phoneme semantics by Pinyin*



请输入要查找的信息的拼音或者字母

shan

单又念单，好事又读好事
朴又读朴

Figure 11. *retrieving by user’s intents (a) User input “shan” to search the surname “单” (b) the surname “单” has been located by a vertical line*

The first “单”, pronounced “dan”, meaning “singular”; the second “单” (the fourth character), pronounced “shan”, a surname. These are author’s intents.

User want to search the surname “单”, by input “shan”, as showed in Fig. 11 (a); the result located at the fourth character as showed in Fig. 11 (b); it is impossible to locate the first character “单” (dan), because it is a different objects. Here, User’s intent matched user’s intent.

A real story about ambiguous word semantics once happened years ago. Two persons, A and B signed a contract as following:

“还欠款 20 万元”， the first character “还” is ambiguous; there are two different pronunciations and two different meaning. It is just the different semantics incurred a lawsuit. In the court,

Person B based the first semantics: A “hasn’t paid ¥20000” to B;

Person A based the second semantics: A “has paid ¥20000” to B”.

The judger no way judged who is right.

Semantic analysis is the foundation of semantic computing [3]; it involves extraction of context-independent aspects of a sentence's meaning. However, even the judger can’t extract the author’s real intents, how an agent can extract the author’s real intents?

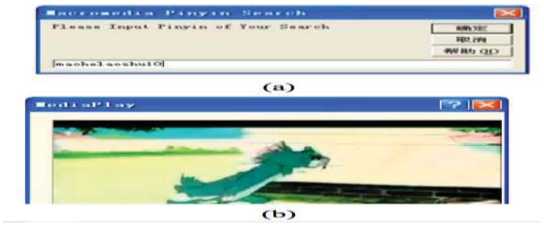
Because Vbit and Vcode have no way to express or store author’s intents precisely, the real intents are often blurred, some of the real intents can be extracted, but some can’t. Therefore, the ability of semantic analysis can’t be overvalued.

However, with this interface, there is only one object of two word objects in the contract; therefore, the meaning is precisely, no ambiguity in the contract, with the real intents; therefore, this kind of lawsuit would never happen again.

7.4 Retrieve any kinds of objects

Example 1, Retrieve one video object: “maohelaoshu10” as shown in Fig. 12.

Example 2, Retrieve image object as shown in Fig. 13.



Please Input Pinyin of Your Search

maohelaoshu10

Figure 12. *(a) Search video: “cat and mouse 10”, (b) play the searched video*

Figure 13. *Image object searched, and displayed.*

7.5 Retrieve the objects selected

User can select the objects by user's intent, then click "Play selected objects", then all selected objects plays one by one, for video object, play the video; for audio object, play the audio; for image, display the image; for text, read the text by text to speech.

8 THE ARCHITECTURE OF USC

The architecture of USC is shown in Fig. 14.

The contents consist of every kinds of objects which are represented in different kinds of Ucodes. There exist contents which are represented in Vcodes, which can be represented by Ucodes in processing.

From precisely view, the contents can be divided into two classes: one is in Ucodes with precisely author's intents; another is in Vcodes with author's blurred intents, called as raw contents.

From security view, contents can be divided into two classes: public contents and private contents.

The interface of USC consists of various Ucode programs, by which author can create contents based on author's intents; user can retrieve contents based on user's intents; manager can manage contents. Anyone could be author, user or manager.

Because the Ucode programs are machine independent, platform independent and network independent, and Ucodes are space consistent and time consistent, contents can be anything, can be processed in anywhere, created, modified, retrieved or managed by anyone in anytime.

Managements of contents can be:

- Convert raw contents by semantic analysis
- Management of Interface tools
- Security management

…

9 CONCLUSION

USC can make computer scientists' dream become true; however, this paper is very elemental; this is just the beginning of USC.

It is the right time to shift VSC to USC; with more scientists taking part in, a bright future is in front of us.

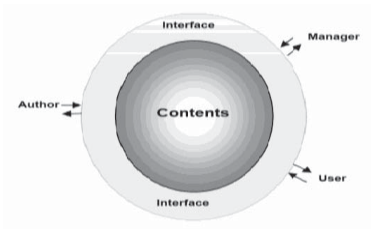


Figure 14. *The architecture of USC*

Acknowledgments Thank Bin Wu, my wife, for her great contributions and fully supporting.

REFERENCES

[1] J. Glenn Brookshear, Computer Science: An Overview (11th Edition), Pearson Higher Education, 2012

[2] Tim Berners-Lee, http://www.techrepublic.com/article/an-introduction-to-tim-berners-lees-semantic-web/

[3] IEEE Press Editorial Board, Lajos Hanzo, Phillip C.-Y., Sheu Heather Yu, C. V. Ramamoorthy Arvind K. Joshi, Lotfi A. Zadeh, SEMANTIC COMPUTING, IEEE Press, the Institute of Electrical and Electronics Engineers, Inc. 2010

[4] Shengyuan Wu, Methods and apparatuses of digital data processing, PCTIB2013060369, 11, 2013

[5] Shengyuan Wu, Introduction to Multilevel Mark Coding Theory, Proceedings of The 2007 International Conference on Foundations of Computer Science (FCS'07), June 2007

[6] Aho, Alfred V. Lam, Monica S, Compilers Principles, Techniques and Tools, New Delhi Pearson, 2011, P 1,1 ,1, 99, 522, 382-385

[7] William Stallings, Cryptography and Network Security principles and Practice, Fifth Edition, Pearson Education, Inc., 2011

[8] Gilles van Assche, Quantum Cryptography and Secret-Key Distillation, Cambridge University press, 2006

[9] Yin Hao, Han Yang, The principles and technology of quantum communication, Electronic Industry Press, 2013

[10] Shengyuan Wu, One-time Pad Cipher Based on Out-Key Distribution, Proceedings of The 2014 International Conference on Wireless Networks of Computer Science (ICW'14), 07, 2014