Semantic representation of events in 3D animation

Minhua Ma and Paul Mc Kevitt
School of Computing & Intelligent Systems
Faculty of Informatics
University of Ulster, Magee
Derry/Londonderry BT48 7JL, Northern Ireland.
E-mail: {m.ma,p.mckevitt}@ulster.ac.uk

Abstract

There is a need for semantic representations that can bridge the gap between linguistic inputs verbs and their corresponding visual knowledge which are indispensable in performing a variety of tasks involving the automatic generation of 3D animation. The semantic representation of events in visual knowledge and the design of a suitable knowledge base specifically for the integration of linguistic and visual information are discussed here. We describe a framework used to represent action verb semantics in a visual knowledge base. Visually observed events are described by establishing a correspondence between verbs and the visual depictions they evoke. The method proposed here is well-suited to practical applications such as automatic language visualisation applications and intelligent storytelling systems. In particular, it will be useful within CONFUCIUS, a system which receives input natural language stories and presents them with 3D animation, speech, and non-speech audio.

1 Introduction

Most traditional semantic representation languages in natural language processing represent meaning on the sentence level or phrase level, and are used for purposes like question-answering and information retrieval. Here we introduce a framework which can represent procedural semantics on the word level for action verbs and is suited for computer graphic generation from natural language input. The framework extends traditional predicate-argument structures down to the verb level and uses a hierarchy of actions to describe visual semantics of action verbs. First, we begin by introducing the background of this work, the *Seanchaí*[1] platform and its *CONFUCIUS* module that interprets natural language story input and presents it as 3D animation with other modalities (section 2). Then we explore previous ways to represent semantics, focusing on conceptual dependency and event-logic (section 3). We turn next to the topic of semantic representation in a knowledge base for natural language and vision processing systems and discuss verb classification from the prospect of the generation of 3D animation (section 4). Next we propose a framework of semantic representation of events, which extends traditional semantic predicate-argument structures from sentence level to word level, giving definitions of verbs and verb phrases to facilitate automatic language visualisation (section 5). Next, relations of our method to other work are considered (section 6) and finally section 7 concludes with a discussion of possible future work on semantic representation of events.

2 Seanchaí and CONFUCIUS

The long-time goal of this work is using the methodology presented here to generate 3D animation automatically in an intelligent multimedia storytelling platform called Seanchaí. Seanchaí

[1] Seanchaí means 'Storyteller' in the Gaelic language.

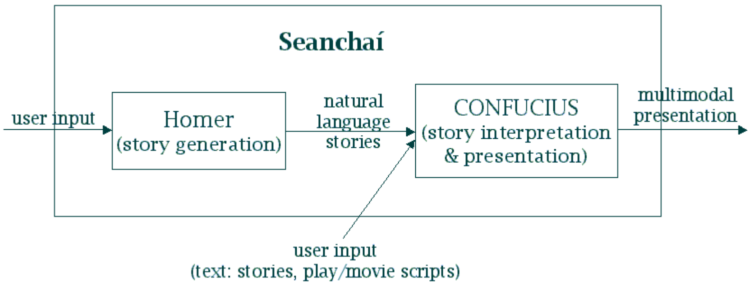

Figure 1: Intelligent multimodal storytelling platform – *Seanchaí*

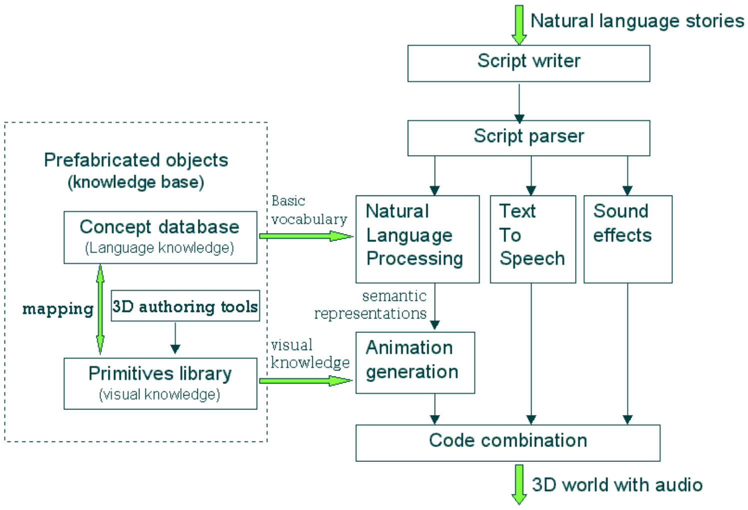

Figure 2: System architecture of CONFUCIUS

will perform multimodal storytelling generation, interpretation and presentation and consists of *Homer*, a storytelling generation module, and CONFUCIUS, a storytelling interpretation and presentation module (Figure 1). The output of the former module could be fed as input to the latter. Homer focuses on natural language story generation. It will receive two types of input from the user, (1) either the beginning or the ending of a story in the form of a sentence, and (2) stylistic specifications, and outputs natural language stories; and CONFUCIUS focuses on story interpretation and multimodal presentation. It receives input natural language stories or (play/movie) scripts and presents them with 3D animation, speech and non-speech audio.

The knowledge base and its visual knowledge representation presented here are used in CONFUCIUS (Figure 2), and they could also be adopted in other vision and natural language processing integration applications. The dashed part in the figure includes the prefabricated objects such as characters, props, and animations for basic activities, which will be used in the *Animation generation* module. When the input is a story, it will be transferred to a script by the *script writer*, then parsed by the *script parser* and the *natural language processing* module respectively. The modules for *Natural Language Processing (NLP)*, *Text to Speech (TTS)* and *sound effects* operate in parallel. Their outputs will be fused at *code combination*, which generates a holistic 3D world representation including animation, speech and sound effects. NLP will be performed using Gate and WordNet, TTS will be performed using Festival or Microsoft Whistler, VRML (Virtual Reality Modelling Language) will be used to model the story 3D virtual world, and visual semantics is represented using a Prolog-like formalism proposed in this paper.

3 Previous semantic representation languages

There are several semantic representation languages which have been implemented for natural language processing: FOPC (First Order Predicate Calculus), semantic networks, Conceptual Dependency (CD) (Schank 1973), and frames (Minsky 1975). FOPC and frames have historically been the principal methods used to investigate semantic issues. Recent research on semantic representation of simple action verbs includes event-logic (Siskind 1995) and x-schemas with f-structs (Bailey et al. 1997). In addition, XML as a mark-up language is used to represent semantic structure in recent multimodal systems, such as in BEAT (Cassell et al. 2001) and a derivative, M3L (MultiModal Markup Language), in SmartKom (Wahlster et al. 2001). The traditional semantic representations (FOPC, semantic networks, CD and frames) are used at sentence/phrase level, e.g. predicate-argument models list as many arguments as are needed to incorporate all the entities associated with a motion, such as `give(sub, indirectObj, directObj)` and `cut(sub, obj, tool)`, while both event-logic and x-schemas work on the word level (action verbs). However, Schank's CD theory also provides fourteen primitive actions to represent and infer verb semantics, i.e. at the word level.

There are many natural language and vision processing integration applications developed based on these semantic representations, such as CHAMELEON (Bröndsted et al. 2001) and WordsEye (Coyne & Sproat 2001) based on frame representations, SONAS (Kelleher et al. 2000) based on predicate-argument representations, SmartKom and BEAT based on XML to represent semantic structure of multimodal content, and Narayanan's language visualisation system (Narayanan et al. 1995) based on CD, ABIGAIL (Siskind 1995) and the L_0 project (Bailey et al. 1997) based on their respective semantic representations. Now we will discuss CD, event-logic truth conditions, and x-schemas in detail since these focus most on semantic representation of action verbs.

3.1 Schank's Conceptual Dependency theory and scripts

Schank (1973) discusses *Conceptual Dependency (CD)* theory to represent concepts acquired from natural language input. The theory provides fourteen primitive actions and six primitive conceptual categories. These primitives can be connected together by relation and tense modifiers to describe concepts and draw on inferences from sentences. CD theory makes it possible to represent sentences as a series of diagrams depicting actions using both abstract and real physical situations. The agents and the objects in the sentences are represented. The process of splitting the knowledge into small sets of low-level primitives makes the problem solving process easier, because the number of inference rules needed is reduced. Therefore CD theory could reduce inference rules since many inference rules are already represented in CD structure itself. However, knowledge in sentences must be decomposed into fairly low level primitives in CD, therefore representations can be complex even for relatively simple actions. In addition, sometimes it is difficult to find the correct set of primitives, and even if a proper set of primitives are found to represent the concepts in a sentence, substantial inference is still required.

Additionally, since people have routines-routine ways of responding to greetings, routine ways to go to work every morning, and so on-as should an intelligent knowbot, Schank introduced *scripts*, expected primitive actions under certain situations, to characterize the sort of stereotypical action sequences of prior experience knowledge within human beings' common sense which computers lack, such as going to a restaurant or travelling by train. A script could be considered to consist of a number of slots or frames but with more specialised roles. The components of a script include: entry conditions, results, props, roles, tracks and scenes.

An implemented text-to-animation system based on CD primitives (Narayanan et al. 1995) shows the limitations of CD. The graphic display in the system is iconic, without body movement details because CD theory focuses on the inferences of verbs and relations rather than the visual information of the primitive actions.

We discuss now two verb semantic representations, event-logic truth conditions and f-structs. Both of them are mainly designed for action recognition from visual input. The goal of our work is a reverse process to visual recognition, i.e. language visualisation in CONFUCIUS. A common problem in the tasks of both visual recognition and language visualisation is to represent visual semantics of motion events, which happen both in the space and time continuum.

3.2 Event-logic truth conditions

Traditional methods in visual recognition segment a static image into distinct objects and classify those objects into distinct object types. Siskind (1995) describes the ABIGAIL system which focuses on segmenting continuous motion pictures into distinct events and classifying those events into event types. He proposed event-logic truth conditions for simple spatial motion verbs' definition used in a vision recognition system. The truth conditions are based on the spatial relationship between objects such as support, contact, and attachment, which are crucial to recognition of simple spatial motion verbs. According to the truth condition of the verbs' definition, the system recognizes motions in a 2D line-drawing movie. Siskind introduced a set of perceptual primitives that denote primitive event types and a set of combining symbols to aggregate primitive events into complex events. The primitives are composed of three classes: time independent primitives, primitives determined from an individual frame in isolation, and primitives determined on a frame-by-frame basis. Using these primitives and their combinations, he gives definitions of some simple motion verbs and verifies them in ABIGAIL.

Siskind's event-logic definition has two deficiencies: (1) overlapping between primitive relations, and (2) lack of conditional selection, i.e. this framework does not provide a mechanism for selection restrictions of the arguments. So some definitions are arbitrary to some degree. They do not give a necessary and sufficient truth-condition definition for a verb. For example: the definitions for 'jump' and 'step' are the following:[2]

$$\begin{aligned} jump(x) &= supported(x); (\neg\Diamond supported(x) \wedge translatingUp(x)) \\ step(x) &= \exists y(part(y,x) \wedge [contacts(y, ground); \neg\Diamond contacts(y, ground); \\ &\quad contacts(y, ground)]) \end{aligned}$$

The definition of 'jump' means x is supported, and then not supported *and* moves up in the immediate subsequent interval. The definition of 'step' can be interpreted that there exists y, which could be a foot, which is part of the x, *and* y first contacts ground, then does not contact, and finally contacts ground again. From the two definitions, we see that the definition of 'step' can also define the motion of 'jump' or 'stamp (a foot)'. Hence, the definition of one verb can also be used to define other verbs. Also, an alternative definition of 'step' based on Siskind's methodology could be:

$$\begin{aligned} step(x) &= \exists y1, y2(part(y1,x) \wedge part(y2,x) \wedge \\ &\quad [(contacts(y1, ground) \wedge \neg\Diamond contacts(y2, ground)); \\ &\quad (\neg\Diamond contacts(y1, ground) \wedge \neg\Diamond contacts(y2, ground)); \\ &\quad contacts(y1, ground)]) \end{aligned}$$

The definition describes the alternate movement of two feet y1 and y2 contacting the ground in a step. Hence, one verb can be defined by many definitions. Siskind's visual semantic representation method is subject to ambiguity, i.e. a single verb can legitimately have different representations such as 'step', and a single representation can correspond to different events such as the first definition of 'step' can define 'jump' and 'stamp' as well. This arbitrariness in the event definition causes some false positives and false negatives when ABIGAIL recognises motions in animation. The deficiency of conditional selection causes some loose definitions, admitting many false positives, e.g. the definition of 'jump' admits unsupported upward movement of some inanimate objects like ball or balloon, because it does not have any semantic constraints

[2]a;b means event b occurs immediately after event a finishes. $\Diamond a@i$ means a happens during i or a subset of i, so $\neg\Diamond supported(x)@i$ means 'x is not supported in any time during i'.

on the fillers of argument x, indicating that x should be an animate creature (non-metaphor usage).

The arbitrariness of verb definition might arise from two problems in his primitives. One is the overlapping between some primitives in individual frame class, such as *contacts()*, *supports()*, and *attached()*. For instance, when one object is supported by another, it usually contacts the supporting object. The other problem is that some primitives in frame-by-frame class are not atomic, i.e. could be described by combinations of others, such as *slideAgainst(x,y)* might be performed by $translatingTowards() \wedge supports(y,x)$. In his methodology, Siskind does not consider internal states of motions (e.g. motor commands), relying instead on visual features alone, such as support, contact, and attachment. This works in vision recognition programs. However, internal states are required in vision generation applications. X-schemas (eXecuting-schema) and f-structs (Feature-structures) (Bailey et al. 1997) examine internal execute motor actions.

3.3 X-schemas and f-structs

Bailey et al.'s (1997) x-schemas and f-structs representation combines schemata representation with fuzzy set theory. It uses a formalism of Petri nets to represent x-schemas as a stable state of a system that consists of small elements which interact with each other when the system is moving from state to state, and each sense of a verb is represented in the model by a feature-structure (f-struct) whose values for each feature are probability distributions. For instance, the f-struct of one word-sense of *push*, using the *slide* x-schema, consists of motor parameter features and world state features. Motor parameter features concern the hand motion features of the action push, which invoke an x-schema with corresponding parameters, such as direction, elbow joint motion, and hand posture. World state features concern the object that the action is performed on, such as object shape, weight, and position.

The probabilistic feature values in this structure are learned from training data. The application based on this representation is a system trained by labelled hand motions which learns to both label and carry out similar actions with a simulated agent. It can be used in both verb recognition and performing the verbs it has learned. However, the model requires training data to create the f-structs of verbs before it can recognize and carry them out.

Most of previous semantic representations are suited to certain purposes. FOPC is good for query-answering (especially a true or false judgement); event-logic truth conditions are suitable for motion recognition; x-schemas with f-structs are suited to both verb recognition and performing the action, but require training.

4 Visual knowledge representation

Existing multimodal semantic representations within various intelligent multimedia systems may represent the general organisation of semantic structure for various types of inputs and outputs and are usable at various stages such as media fusion and pragmatic aspects. However, there is a gap between high-level general multimodal semantic representation and lower-level representation that is capable of connecting meanings across modalities. Such a lower-level meaning representation, which links language modalities to visual modalities, is proposed in this paper. Figure 3 illustrates the multimodal semantic representation of CONFUCIUS. It is composed of language, visual and non-speech audio modalities. Between the multimodal semantics and each specific modality there are two levels of representation: one is a high-level multimodal semantic representation which is media-independent, the other is an intermediate level media-dependent representation. CONFUCIUS will use an XML-based representation for high-level multimodal semantics and an extended predicate-argument representation for intermediate representation which connects language with visual modalities as shown in Figure 3. The visual semantics decomposition method discussed here is at the intermediate representation level. It is suitable for implementation in the 3D graphic modelling language VRML. It will be translated to VRML

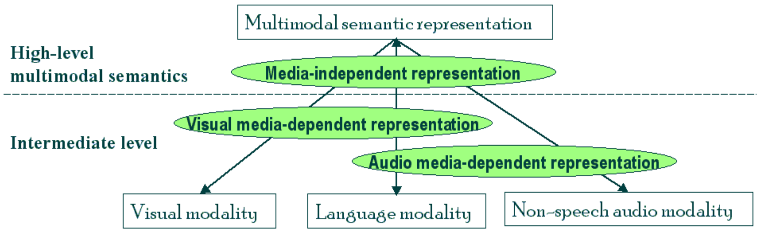

Figure 3: MultiModal semantic representation in CONFUCIUS

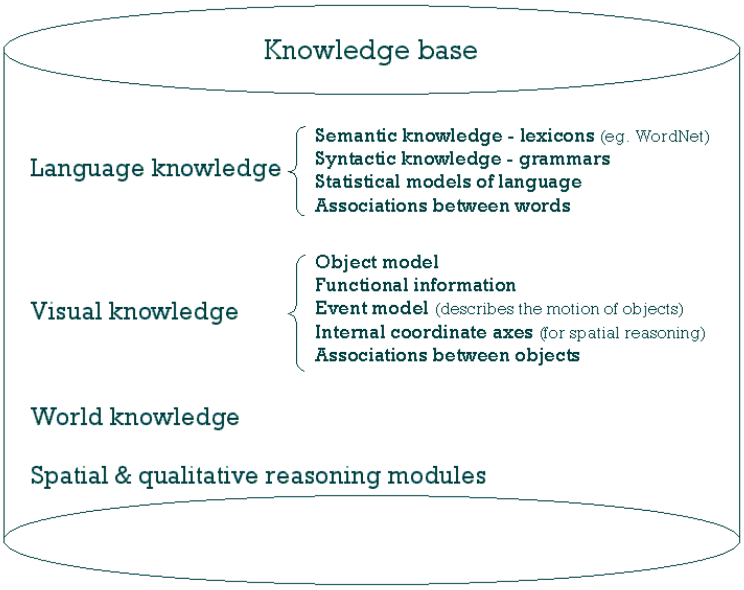

Figure 4: Knowledge base of CONFUCIUS

code by a Java program in CONFUCIUS. We also plan to include non-speech audio in the media-dependent and media-independent semantic representations.

4.1 Visual semantics

The knowledge representation required for CONFUCIUS must provide the following capabilities: (1) model both declarative and procedural knowledge, (2) inference mechanisms for object classification, lexical dependencies, and spatial relations. Figure 4 illustrates the contents of the knowledge base of CONFUCIUS, and is also a general knowledge base design for any intelligent multimedia applications which include both natural language processing and vision processing. This knowledge base is the prefabricated objects, dashed part of Figure 2. It consists of language models, visual models, world knowledge and spatial & qualitative reasoning modules. Language knowledge is used in the natural language processing component (shown in Figure 2) to extract concept semantics from text. Visual knowledge consists of the information required to generate moving image sequences. It consists of object model, functional information, event model, internal coordinate axes, and associations between objects. The Object model has semantic representation of categories (nouns), the event model has semantic representation of motions (verbs), and the internal coordinate axes are indispensable in some primitive actions of event models such as rotating operations, which require spatial reasoning based on the object's internal axes.

Here we focus on efficient semantic representation of event models in the typical knowledge base for natural language and vision integration systems. The event model in visual knowledge requires access to other parts of visual knowledge. For instance, in the event "he cut the cake", the verb "cut" concerns kinematical knowledge of the subject-human being, i.e. the movement of his hand, wrist and forearm. Hence it needs access to the object model of a man who performs the action, "cut". It also needs function information of "knife", the internal coordinate axes information of "knife" and "cake" to decide the direction of the movement. To interpret the verb *wear(x,y)*[3], the event model needs access to the object model of y, which might be a "hat", a "ring", a "pair of glasses", or "shoes", and its function model that concerns its typical location (e.g. "hat on the head", "ring on a finger").

4.2 Categories of events in animation

Since verbs are core to events, verb subcategories are significant for visualisation of events. Both traditional grammars subcategorising verbs into transitive and intransitive, and modern grammars distinguishing as many as 100 subcategories (COMLEX and ACQUILEX tagsets) classify verbs according to subcategorisation frames, i.e. possible sets of complements the verbs expect (Table 1). Here we subcategorise verbs in animation from the visual semantic perspective, as shown in Figure 5, albeit the classification has overlays with linguistic subcategorisation.

Frame	*Verb*	*Example*
Ø	eat, sleep	I want to eat
NP	prefer, find, leave	find [$_{NP}$ the flight from New York to Boston]
NP NP	show, give	show [$_{NP}$ me] [$_{NP}$ airlines with flights from New York]
PP_{from} PP_{to}	fly, travel	I would like to fly [$_{PP}$ from New York] [$_{PP}$ to Boston]
VP_{to}	prefer, want, need	I want [$_{VPto}$ to have a pint of beer]
$VP_{bareStem}$	can, would, might	I can [$_{VPbareStem}$ swim]
S	mean, say, think, believe	He said [$_S$ the Government disagreed with her account]

Table 1: Some linguistic subcategorisation frames and example verbs

First we divide all events into events on atomic entities and events on non-atomic entities based on whether the objects they act on have sub-components or not. Non-atomic entities are constructed out of collections of objects. In the atomic entities group, we classify the events to those changing objects' physical location and those changing objects' intrinsic attributes like shape, size, and colour. In the non-atomic group, we classify them according to whether they concern a character. Those events based on characters can next be divided based on whether or not they are easily observable. Action verbs are easily observed and hence the major part of events in animation. Action verbs can be further classified into transitive verbs and intransitive verbs, according to their accusative nature, i.e. if the action concerns object(s). In the action verb group, the movement of the subject which performs the action usually could be modelled by biped kinematics, and the movement (or change) of the objects or tools (for transitive verbs) could be modelled by event models. In the intransitive verb group, there is an important class involving multimodal presentation – communication verbs. These verbs require both visual presentation such as lip movement (e.g. "speak", "sing"), facial expressions (e.g. "laugh", "weep") and audio presentation such as speech or other communicable sounds.

Verbs working on atomic entities can also work on non-atomic objects, for instance, "disappear/vanish" can work on both atomic and non-atomic objects (the Cheshire cat in the following example 1) or component(s) of non-atomic objects like "tail" in example 1 and "head" in example 2.

(1) 'All right,' said the Cat. And this time it vanished quite slowly, beginning with the end of its tail and ending with its grin.

(2) He turned his head and looked back.

[3] *wear(x,y)* means x wears y; x is a person or personated character, y is an object.

- **Atomic entities**
  - *Change physical location such as position and orientation, e.g. bounce, turn*
  - *Change intrinsic attributes such as shape, size, color, texture, e.g. bend, taper, and even visibility, e.g. disappear, fade in/out*
- **Non-atomic entities**
  - *Non-character events*
    - Two or more individual objects fuse together, e.g. melt (in)
    - One objects divides into two or more individual parts, e.g. break (into pieces), (a piece of paper is) torn (up)
    - Change sub-components (their position, size, color, shape etc), e.g. blossom
    - Environment events (weather verbs), e.g. snow, rain, gettingDark
  - *Character events*
    - **Action verbs**
      - Intransitive verbs
        - *Biped kinematics, e.g. walk, swim, and other motion model like fly*
        - *Face expressions, e.g. laugh, angry*
        - *Lip movement, e.g. speak, say, sing*
      - Transitive verbs
        - *Concern single object, e.g. throw, push, kick*
        - *Concern multiple objects*
          - Transitive verbs with direct and indirect objects, e.g. give, pass, show
          - Transitive verbs with indirect object & the tool used to perform the action, e.g. cut, peel
    - **Non-action verbs**
      - stative verbs (change of state), e.g. live, die, sleep, wake
      - emotion verbs, e.g. like, disgust
      - possession verbs, e.g. have, belong
      - mental activities, e.g. decide, believe, imagine, know, think
      - cognition & perception, e.g. listen, watch, hear, see
    - **Idioms & metaphor verbs**

Figure 5: Categories of events

Non-action verbs and verbs with metaphor meanings are not easily observable. They are hard to describe by physical changes and however are common in even simple stories like children's stories. In performance art, they are often expressed by face expressions, body poses, or more straightforward, through the speech modality. Static language visualisation is used to express mental activities with thinking bubbles. The visual semantics of this type of visualisation is beyond the scope of this paper.

Since the tasks of representing transitive verbs can be divided into two sub-tasks: modelling biped kinematics for the subject, and modelling atomic entity changes for objects and tools, representing kinematics becomes the main task in visual models. There are two ways to describe a biped's kinematical motion. One method is to model different body parts as distinct objects in object modelling (see Figure 4), e.g. forearm and upper arm, leg and shin are individual objects which attach together, event modelling then transforms these individual objects (body parts) that the motion concerns using some kinematical simulators. This method requires substantial work in event modelling in order to achieve realistic effects. The second method, which is also in vogue, regards the body as a whole non-atomic object consisting of several sub-components which attach to a skeleton system in object modelling. Event modelling moves the bone (as a parent or a child in the hierarchical skeleton tree) and passes the movement to its attached parents or children using forward kinematics or inverse kinematics. Forward kinematics is a system in which the transforms of the parent in a hierarchical tree structure are passed on to the children, and animation is performed by transforming the parent. Inverse kinematics is a system in which the movement of the children is passed back up the chain to the parent. Animation is performed by affecting the ends of the chain, e.g. in biped walking animation, by moving the foot and the shin, knees and thighs rotate in response. Inverse kinematics models the flexibility and possible rotations of joints and limbs in 3D creatures. This approach gives the best possibilities to produce life-like animation of a character in a story, but does involve effort to set up the object models for the characters.

5 Extending predicate-argument representation to word level

In this section, we will extend the predicate-argument representation of verbs from phrase/sentence level to word level in a bottom-up fashion by first examining primitives and then showing how they can be composed to define composite event semantics. The discussion of semantic representation of object models is beyond the scope of this paper. The extended predicate-argument model describes verbs and verb phrases. To define objects, their properties and categories is the task of the object model (in visual knowledge), which may declare an object as an instantiation of a type (its category membership) at first, and give its attributes. The common attributes are *position*, *orientation*, *size* and so on. Some complex objects may have other attributes like *gender*, *age* and so on, for an instance of *person*. For example, *Alice* (a six years old girl) is an instance of *person*, with common attributes like her initial *location*, *orientation* (face to which direction), *height* (size) and specific attributes such as *female* (gender), and *6* (age) as well. The body parts are defined in the prototype of *person*. The extended predicate-argument model is aimed at automatic generation of animation from linguistic input (language visualisation). It can also be used to expand FOPC representations for application to lower levels of linguistic input.

5.1 Constants, variables and types

There are a few constants in this framework referring to specific objects which exist in every scene of the virtual world. We use the convention that names of constants in CONFUCIUS are composed of capitals and underscores. `GROUND` is a plane in the coordinates `(0,0,0)` with the length and width which are greater than the space of the visualised scene and has the function of supporting things which otherwise will look like they were floating in the air. `SUN` is the default ambient light which illuminates objects in the scene. The term constant in CONFUCIUS' knowledge base has a different sense than that used in programming languages, i.e. its values can be changed. The are constants in the sense of their constant existence in the simulated visual world. For example, though it has infinite value for its length and width, one can change the size of the `GROUND` according to the size of the stage for facilitating implementation. Similarly, for `SUN`, when the language input describes that "It turned dark." or "in dusk", the brightness of `SUN` can be changed.

Unlike in FOPC, some proper nouns such as *Mary* (person's name) are not treated as constants in the framework but *variables* (instances of the type *Man/Woman*). Like in other programming languages, variables in CONFUCIUS can denote names of objects, which is an instance of a type. A variable name is started from a lowercase letter and can be followed by letters (uppercase or lowercase), numbers, underscores and hyphens. Object parts and properties can be referred to by a dot operator, e.g. `alice.righthand` or `alice.height`.

Type is the name of a category. We use the convention that a type name begins with a capital and is followed by letters, numbers, underscores or hyphens. As in WordNet (Beckwith et al. 1991), objects inherit all the properties of their super-concepts (parents). However, attributes such as size, color, position, and so on can be specialised. There are two operations involving type: `type(objName, typeName)` and `aKindOf(subtypeName, parentTypeName)`, e.g. `type(alice, Girl)`, `aKindOf(Girl, Person)`.

5.2 Hierarchical structure of predicate-argument primitives

The predicate-argument format we apply to represent verb semantics has a Prolog-inspired nomenclature. Each non-atomic action is defined by one or more subgoals, and the name of every goal/subgoal reveals its purpose and effect. Primitives 1 through 14 below are basic primitive actions in our framework (Figure 6). We do not claim that these fourteen cover all the necessary

```
 1) move(obj, xInc, yInc, zInc)
 2) moveTo(obj, loc)
 3) moveToward(obj, loc, displacement)
 4) rotate(obj, xAngle, yAngle, zAngle)
 5) faceTo(obj1, obj2)
 6) alignMiddle(obj1, obj2, axis)
 7) alignMax(obj1, obj2, axis)
 8) alignMin(obj1, obj2, axis)
 9) alignTouch(obj1, obj2, axis)
10) touch(obj1, obj2, axis)  ; for the relation of support and contact
11) scale(obj, rate)  ; scale up/down, change size
12) squash(obj, rate, axis)  ; squash or lengthen an object
13) group(x, [y|_], newObj)
14) ungroup(xyList, x, yList)
```

Figure 6: Basic predicate-argument primitives within CONFUCIUS

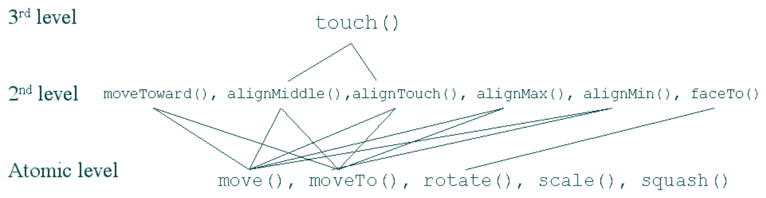

Figure 7: Hierarchical structure of CONFUCIUS' primitives

primitives needed in modelling observable verbs. 13[4] and 14[5] are actually not primitive actions, but they are necessary in processing complex space displacement. In the first twelve primitives, 1-3 describe position movement, 4 and 5 concern orientation changes, 6-9 focus on alignment, 10 is a composite action (not atomic) composed by lower level primitives, and 11, 12 concern size (shape) changes. Figure 7 illustrates the hierarchical structure of the twelve primitives. Higher level actions are defined by lower level ones. For instance, alignment operations are composed by `move()` and/or `moveTo()` predicates. We will explain these primitives below.

`move(obj, xInc, yInc, zInc)` moves `obj` by designated displacement on the specific axis (axes). Arguments `xInc, yInc`, and `zInc` can be a place-holding underscore (or zero) indicating no displacement on the axis. For example, `move(glasses,_,10,_)` means 'move the glasses up 10 units'.

`moveTo(obj, loc)`, as shown below, moves `obj` to a specific `position` designated by `loc` which is an instance of type `Position`, consisting of 3D coordinates.

```
moveTo(obj, loc):-
    type(loc, Position).
```

`moveToward(obj, loc, displacement)`, as shown below, moves `obj` towards/away from a designated `position` by some `displacement`. The second argument is 3D coordinates of the destination. The third argument can be positive, which means 'move toward the destination'; or negative, which means 'move away from the destination'. If the third argument is uninstantiated, when called `moveToward(obj,loc,displacement)`, the displacement will be a positive random

[4]As is the convention in the programming language Prolog, arguments can be replaced by an underscore if they are undetermined.

[5]ungroup element x from a list which contains it. yList is the rest of the list after deleting x from the original list. This is also a basic list operation in Prolog.

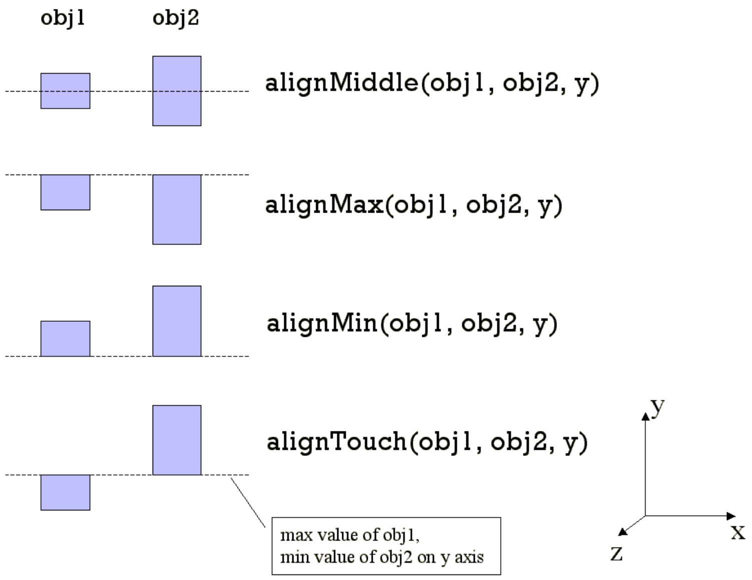

Figure 8: Align predicates (on y axis) in a 3D coordinate system

value greater than 0 and less than the distance between `obj` and destination `location`. This is a second level action, implemented by movement primitive actions at the first level (Figure 7).

```
moveToward(obj, loc, displacement):-
type(loc, Position).
```

`rotate(obj, xAngle, yAngle, zAngle)` rotates `obj` on the designated internal axis (axes). The last three arguments are not external absolute coordinate axes, but the internal coordinate axes of the `obj`, which is defined in visual knowledge/internal coordinate axis (Figure 4). It will be shown with CONFUCIUS that using internal axes is more practical than external axes.

`faceTo(obj1, obj2)` is a second level action, also involving an object's internal coordinate axes. It faces `obj1` to `obj2`. This operation concerns not only `obj1`'s internal axes, but also its functional information (Figure 4) in which its face is defined. For example, in `faceTo(book, alice)`, the book is a cube with face is defined as the book cover. This action rotates this book on its internal axes to make its cover face to `alice`.

Figure 8 illustrates alignment actions from 6 through 9. Note that,
`alignTouch(obj1, obj2, axis)`
uses the first object's maximum value and the second object's minimum value along the axis.

`touch(obj1, obj2, axis)` is a composite action on the third level in the primitive hierarchy. It moves the first argument to the destination where it can touch the nearest face of `obj2`. Figure 9 shows how `touch()` behaves given different axes as the third argument. The comma separating each sub-activity indicates the sequential temporal relation. `act01, act02` means that `act02` begins immediately after `act01` is completed. In addition, ';' is used to indicate the temporal relation between several activities which occur at the same time. `act01; act02` means that `act01` and `act02` begin and end at the same time, i.e. the durations of the two activities are equal. This temporal relation is usable for defining verbs such as "roll" in for example "rolling of a wheel".

```
roll(obj, rollingAngle, newPosition):-
        moveTo(obj, newPosition);
        rotate(obj, 0, 0, rollingAngle).
```

`scale(obj, rate)` scales up/down the size of the object according to the specified rate. `rate` is a real number indicating how much the size should be changed, 1.0 means the same size as the

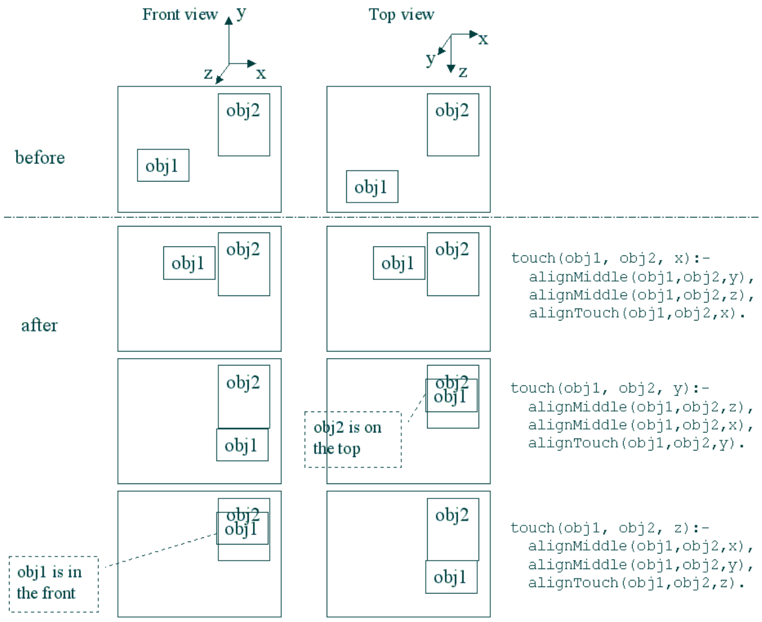

Figure 9: touch() along different axes

original, 2.0 means double size, and 0.5 means half size, on all three axes. `squash(obj, rate, axis)` changes both size and shape of `obj` vis-à-vis `scale()`. It only changes the size on the specified axis. For instance, `squash(sphere01, 0.8, y)` means "squash the ball called sphere01 to 0.8 of its original size on its y axis". `Squash()` can perform more realistic visual effects on elastic objects like balls.

5.3 Visual definitions in extended predicate-argument model

Using predicate-argument primitives we can provide definitions of visual semantics of verbs, investigate how they relate to word senses and consider special families of verbs.

5.3.1 Example visual definitions

Given below are some examples of visual semantics of verbs using the above primitives:

Example 1, *jump*[6] :

```
jump(x):-
        type(x, Animal),
        move(x.feet, _, HEIGHT, _),
        move(x.body, _, HEIGHT, _),
        move(x.feet, _, -HEIGHT, _).
```

Other complex joint movements will be modelled by inverse kinematics. If the second subgoal about movement of the man's body is missing, we will see the man's feet lift with some corresponding movements on his legs (by inverse kinematics), but his body keeps at the original height. Then the reverse subgoal, `move(man, _, -HEIGHT, _)` is not necessary because it will be performed by inverse kinematics automatically.

Example 2, *call*:

– as in "A is calling B" (verb tense is not considered here because it is at sentence level rather than word level). This is one word-sense of call where calling is conducted by telephone.

[6]Semantic constraint – declare an instance of the type 'Animal'. Metaphor usage of vegetal or inanimate characters is not considered here.

Here is the definition of one word-sense of call which is at the first level of the visual semantic verb representation hierarchy:

```
call(a):-
        type(a, Person),
        type(tel, Telephone),
        pickup(a, tel.receiver, a.leftEar),
        dial(a, tel.keypad),
        speak(a, tel.receiver),
        putdown(a, tel.receiver, tel.set).
```

Note there is only one argument for "call" though it is a transitive verb in the example "a is calling b" – but we don't care who b is since he is not present in the scene. The `type(a, Person)` operation is a semantic constraint, to declare that a is an instance of the type `Person`. The variable `tel`, an instance of `Telephone`, is from association between the event `call` and the object `telephone` in the knowledge base (Figure 4). `speak()` involves lip movement and coordinates with another modality – speech. `putdown()` refers to the movement of `hangup`.

Here is the definition of `pickup` which is at the second level of the visual semantic verb representation hierarchy. The three arguments of `pickup()` are *subject*, *object* and *destination* position respectively. `dest` is the 3D coordinates of a location. Here `obj` is a telephone receiver.

```
pickup(x, obj, dest):-
        type(x, Person),
        moveToward(x.leftHand, location(Obj), location(Obj)-location(x)-5),
        touch(x.leftHand, obj, axis),
        group(x.leftHand, obj, xhandObj),
        moveToward(xhandObj, dest, _).
```

Here is the definition of `putdown` which is also at the second level of the visual semantic verb representation hierarchy. The variable x is an aggregation object containing a person and object(s) which are not part of the human body. `obj` is a telephone receiver.

```
putdown(x, obj, dest):-
        moveTo(x.leftHand, dest),
        ungroup(x, obj, x1),
        type(x1, Person).
```

There are many complex issues left unconsidered in the example, such as how to put down the receiver in the exact place of the phone set. However, this can be achieved by some further operations such as a combination of aligns and moves with the calculation of the location of a part of an object. The above examples show a hierarchical representation that involves multiple levels of visual description and the ability to perform top-down interpretation when necessary right down to the primitive level. The implementation of the visual semantic representation provides a backtracking mechanism similar to Prolog which is convenient and efficient. Constructing a complete event semantics of the visual knowledge base (event model in Figure 4) by the above method requires extensive work on verb definitions according to their corresponding semantic knowledge in the lexicon (Figure 4). However, such work is indispensable for an automatic language visualisation system.

Although semantic decomposition of lexical meanings is not a new idea in verb semantic analysis such as Schank's Conceptual Dependency analysis, pros and cons are within the language modality only, and few considerations of visual presentation of verb semantics are given. Jackendoff (1972) advocates semantic decomposition for the generative facilities it provides. He thinks that the decomposition of word meaning into smaller semantic elements allows specification of a generative, compositional system which constrains the way such elements can be related and thereby constrains the ways in which sentences can be constructed, i.e. to prevent semantically anomalous sentences. Opponents of semantic decomposition argue that it is inadequate because a list of necessary and sufficient conditions of a word meaning does not adequately capture the creative aspect of meaning. Linguists have attempted to set forth the full and complete semantic structure of some particular lexical items, but some residue of unexpressed meaning is always left.

Some verb semantic predicates such as "move", "go", or "change" are argued to be the basic components of most verbs from a wide variety of different semantic fields (Jackendoff 1976, Dowty 1979). The decompositional methodology we proposed above differentiates from the previous semantic decomposition theories in two ways: firstly, it is aimed at presentation purposes for visual modalities rather than the generative or interpretative purposes in language modalities; and secondly it does not emphasize atomic predicates. The basic predicates we listed in Figure 6 are not only at the atomic level but also at other higher compound levels and, it is possible to choose predicates at any level to construct a new verb definition.

5.3.2 Visual definition and word sense

The extended predicate-argument model works at the word level, but every definition is for one word sense rather than one word. According to Beckwith's statistics (Fellbaum 1998), one verb has 2.11 senses on average in Collins' English dictionary. For instance, "beat" may have two definitions for the sense of "strike, hit" and the sense of "stir, whisk (cooking verb)". Vice versa, synonyms like "shut" and "close" can share one definition. Disambiguation is a task of the language parser that is solved (probably by selectional restrictions) in language modalities. However, one word sense of a verb may have more than one visual definition because word sense is a minimal complete unit[7] of conception in language modalities whilst visual definition is a minimal complete unit of visual representation in visual modalities. Take the word sense "close" as an example again, there could be three visual definitions for closing of a normal door (rotation on y axis), a closing of a sliding door (moving on x axis), or a closing of a rolling shutter door (a combination of rotation on x axis and moving on y axis).

5.3.3 Troponym verbs

There is a semantic relation in the verb family called "troponym" (Fellbaum 1998). Two verbs have a troponym relation if one verb elaborates the manner of another (base) verb. For instance, "mumble" (talk indistinctly), "trot" (walk fast), "stroll" (walk leisurely), "stumble" (walk unsteadily), "gulp" (eat quickly), the relation between mumble and talk, trot/stroll/stumble and walk, gulp and eat is troponymy. In CONFUCIUS, we use the method of base verb + adverb to present manner elaboration verbs, that is, to present the base verb first and then, to modify the manner (speed, the agent's state, duration of the activity, iteration, and so on) of the activity. To visually present "trot", we create a loop of walking movement, then modify the `cycleInterval` to a smaller number to present fast walking.

5.3.4 Stative verbs

English has productive morphological rule deriving verbs from adjectives or nouns via affixes such as -en, -ify and -ize. These derived verbs, e.g. "whiten", "lengthen", "shorten", "strengthen", "usually" refer to a change of state or property. Those relating to change of size (e.g. "enlarge", "lengthen") could be defined by the predicates `scale()` or `squash()`; those relating to other properties like colour (e.g. "whiten") could be implemented by changing the corresponding property fields of the object in VRML code, which will be discussed in future work on semantic representation of attributes.

[7] Strictly speaking, the smallest meaningful unit in the grammar of a language is morpheme. But morpheme is not a complete unit since it often concerns affix.

5.4 Implementation of extended predicate-argument representation

At the intermediate level visual semantic representation that we have shown in Figure 3, the extended predicate-argument model of motion verbs we proposed above will be parsed by a Java program which finds the 3D object/agent that performs the activity the verb describes in VRML, changes its corresponding keys and key values of interpolators, and hence creates the animation of the object/agent. For example, when creating animation for "A ball is bouncing" (visual definition of "bounce" in Figure 10 (a)), the program looks for the object "ball" in the VRML file (Figure 10 (b)), and creates the bounce animation by adding a timer and position interpolator (with its keys and key values) of the ball (Figure 10 (c)).

5.5 Visual semantic representation of active and passive voice

One important difference between active and passive voice is semantic: the subject of an active sentence is often the semantic agent of the event described by the verb (*He* received the letter) while the subject of the passive is often the undergoer or patient of the event (*The letter* was received), i.e. the topic of active voice is the performer but the topic of passive voice is the undergoer. In CONFUCIUS' visualisation, the semantic difference of voice is represented by point of view, the perspective of the viewer in the virtual world. Since the virtual world in CONFUCIUS is modelled in VRML, *Viewpoint node* is used to represent voices. With the Viewpoint node, one can define a specific viewing location for a scene like a camera. In the previous example, although the two sentences describe the same event, receiving the letter, in active voice the focus is the person who received it, whereas in passive voice it is the letter. Therefore the modelling of the event and concerned object/character are the same for the two sentences, the difference is the parameters (orientation and position) of Viewpoint node to represent the topic in each voice. Besides voice, Viewpoint node may also present converse verb pairs such as "give/take", "buy/sell", "lend/borrow", "teach/learn". They refer to the same activity but from the viewpoint of different participants.

6 Relation to other work

Extended predicate-argument representation can be regarded as an extension of Schank's scripts. Consider the following script of "rob" and "order food" in Figure 11. The extended predicate-argument representation is a script-like rule system from this viewpoint. However, compared with scripts our method uses different primitives to represent visual semantics of events. Table 2 shows a comparison of extended predicate-argument representations and scripts. Both scripts and extended predicate-argument representations translate high level events to lower level events. Level 1 includes routine events (complex activities) that are either lexicalised to verbs (e.g. "interview") or verb phrases (e.g. "eat out", "see a doctor"). Level 2 includes simple action verbs such as "jump", "push". Level 3 includes a finite set of universal semantic components (primitives, atomic actions, or atomic predicates) into which all event verbs/verb phrases could be exhaustively decomposed.

Figure 11 [1] and [3] translate routine events (level 1 in Table 2) to simple action verbs (level 2 in Table 2) whilst [2] and [4] translate simple action verbs to primitive actions (level 2 to level 3 in Table 2). The difference between these two methods is in level 2 to 3 translation. Extended predicate-argument representation focuses on visual presentation of events whilst scripts are suited for language comprehension. According to our events categories in Figure 5, for character events, level 2 to 3 translation of our method describes biped kinematics such as "pick up"; for other events, it interprets the visual semantics of verbs such as "bounce" using primitives (e.g. move, rotate) or changes object properties directly (e.g. "bend", "turn red").

Previous language visualisation systems (Narayanan et al. 1995) built on Schank's CD theory have no image details of dynamic events though this has advantages in inferences from verbs; the world in SONAS (Kelleher et al. 2000) has 3D image details but only concerns simple spatial relation instructions such as "move ten metres forward to the green house" and lacks

```
bounce(obj):-
  move(obj, 0, 20, 0),
  move(obj, 0, -20, 0).
```

(a) visual definition of the verb ‘‘bounce’’ (unaccusative)

```
DEF ball Transform {
  translation 0 0 0
  children [
    Shape {
      appearance Appearance {
        material Material { diffuseColor 0.6 0.8 0.8 }
      }
      geometry Sphere { radius 5 }
    }
  ]
}
```

(b) Input -- VRML code of a static ball

```
DEF ball Transform {
  translation 0 0 0
  children [
  DEF ball-TIMER TimeSensor { loop TRUE cycleInterval 0.5 },
  DEF ball-POS-INTERP PositionInterpolator {
    key [0, 0.5, 1 ]
    keyValue [0 0 0, 0 20 0, 0 0 0 ] },
    Shape {
      appearance Appearance {
        material Material { diffuseColor 0.6 0.8 0.8 }
      }
      geometry Sphere { radius 5 }
    }
  ]
ROUTE ball-TIMER.fraction_changed TO ball-POS-INTERP.set_fraction
ROUTE ball-POS-INTERP.value_changed TO ball.set_translation
}
```

(c) Output -- VRML code of a bouncing ball

Figure 10: 3D animation from predicate-argument representation

```
     Scripts                          Extended predicate-argument representation

[1] Rob(person, place):-              [3]  call(a):-
    obtain(person, gun),                   pickup(a, tel.receiver, a.leftEar),
    go(person, place),                     dial(a, tel.keypad),
    holdUp(person, place),                 speak(a, tel.receiver),
    escape(person, place).                 putdown(a, tel.receiver, tel.set).

[2] OrderFood(person):-               [4]  pickup(x,obj,dest):-
    ATRANS(waiter,person,menu),            moveToward(x.leftHand,location(obj),
    MTRANS(menu, person),                  location(obj)-location(x)-5),
    MBUILD(person, choice),                touch(x.leftHand, obj, y),
    TRANS(person,waiter,choice).           group(x.leftHand, obj, xhandObj),
                                           moveToward(xhandObj, dest, _).
```

Figure 11: Examples of scripts and extended predicate-argument representation

Event levels	*Example verbs*
(1) Routine events, complex activities	Rob, cook, interview, eatOut
(2) Simple action verbs	jump, lift, give, walk, push
(3) Primitive actions	ATRANS, PTRANS, MOVE (Script)
	move, rotate (Extended predicate-argument representation)

Table 2: Comparison of scripts and extended predicate-arguments

collision detection; ABIGAIL (Siskind 1995) is applied to spatial motion verb recognition, but not synthesis, and has ambiguity in some verbs' definition; and L_0 (Bailey et al. 1997) can both label verbs and carry them out but requires training data to construct probabilistic feature-structures of verbs.

7 Conclusion and future work

The methodology introduced in this paper is a visual semantics representation that describes procedural information of action verbs and facilitates automatic animation generation from text. This work is most relevant in the context of automatic generation of 3D animation from linguistic input. The hierarchical extended predicate-argument representation of visual semantics for events may be applied in the knowledge base of automatic text-to-animation applications like CONFUCIUS. The main goal of this work is to develop a suitable methodology for formalizing the meanings of action verbs. Hence we conclude the visual semantics representation for CONFUCIUS is appropriate and will be implemented using Java and VRML where the Prolog-like intermediate representation will be parsed by a Java program. The reusability of the visual semantics representation and the knowledge base of CONFUCIUS is considered. The language and visual knowledge shown in Figure 4 are dependent on one another because entries in the graphic library depend on the taxonomy and granularity of lexicon in the language knowledge base. However, the language and visual knowledge as a whole may be independent from other modules of CONFUCIUS and reused in other language and vision integration systems, and other application domains.

Further work will attempt to solve the following issues. One major problem that text-to-animation applications face is vagueness. Most text-to-graphics systems solve the vagueness in natural language by substituting an object type with a more specific object of the type. For example, to visualise the phrase "give her a toy" by substituting *a toy* with a specific toy such as a teddy bear. Verb semantics has the same problem. Although it can be solved by specific-general substitution, a vague representation of the meaning of the object/event may be appropriate in some situations. It will be advantageous for a semantic representation to maintain a certain level of vagueness. But it is almost impossible in vision, because visual modalities require more specific information than language modalities. The issue of representing temporal information of events is not addressed in this paper, whilst it is important for the information that verb tenses convey. Our current model of predicate-argument decompositional representation describes the order of subactivities but does not indicate how long each subactivity takes. Moreover, future work will also consider visual semantic representation of nouns, non-action verbs, adjectives, adverbs, and prepositions in 3D animation.

References

Bailey, D., J. Feldman, S. Narayanan & G. Lakoff (1997) Modeling embodied lexical development. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society (CogSci97), 19-24*, Stanford, CA, USA.

Beckwith, R., C. Fellbaum, D. Gross and G.A. Miller (1991) WordNet: a lexical database organised on Psycholinguistic principles. In *Lexicons: using on-line resources to build a lexicon*, U. Zernik (Ed.), 211-231. Lawrence Erlbaum: Hillsdale, NJ.

Bröndsted, T., P. Dalsgaard, L.B. Larsen, M. Manthey, P. Mc Kevitt, T.B. Moeslund & K.G. Olesen (2001) The IntelliMedia WorkBench - an Environment for Building Multimodal Systems. In *Advances in Cooperative Multimodal Communication: Second International Conference, CMC'98, Tilburg, The Netherlands, January 1998, Selected Papers*, H. Bunt and R.J. Beun (Eds.), 217-233. Lecture Notes in Artificial Intelligence (LNAI) series, LNAI 2155, Berlin, Germany: Springer Verlag.

Cassell, J., H. Vilhjalmsson and T. Bickmore (2001) BEAT: the behaviour expression animation toolkit. In *SIGGRAPH 2001 Conference Proceedings*, Los Angeles, August 12-17, 477-486.

Coyne, B & R. Sproat (2001) WordsEye: an automatic text-to-scene conversion system. In *AT&T Labs. Computer Graphics Annual Conference, SIGGRAPH 2001 Conference Proceedings*, Los Angeles, Aug 12-17, 487-496.

Dowty, D. (1979) *Word meaning and Montague Grammar.* Dordrecht: Reidel.

Fellbaum, C. (1998) A semantic network of English verbs. In *WordNet: An Electronic Lexical Database*, C. Fellbaum (Ed.), 69-104. Cambridge, MA: MIT Press.

Jackendoff, R.S. (1972) *Semantic interpretation in generative grammar.* Cambridge, Mass.: MIT Press.

Jackendoff, R.S. (1976) Toward an explanatory semantic representation. In *Linguistic Inquiry*, Vol. 7: 89-150.

Kelleher, J., T. Doris, Q. Hussain & S. Ó Nualláin (2000) SONAS: multimodal, multi-user interaction with a modelled environment. In *Spatial Cognition*, S. Ó Nualláin (Ed.), 171-184. Philadelphia, USA: John Benjamins.

Minsky, M. (1975) A Framework for representing knowledge. In *Readings in knowledge representation*, R. Brachman and H. Levesque (Eds.), 245-262. Los Altos, CA: Morgan Kaufmann.

Narayanan, S., D. Manuel, L. Ford, D. Tallis & M. Yazdani (1995) Language visualisation: applications and theoretical foundations of a primitive-based approach. In *Integration of Natural Language and Vision Processing (Volume II): Intelligent Multimedia*, P. Mc Kevitt (Ed.), 143-163. London, England: Kluwer Academic Publishers.

Schank, R.C. (1973) *The fourteen primitive actions and their inferences.* Memo AIM-183, Stanford Artificial Intelligence Laboratory, Stanford, CA, USA.

Siskind, J.M. (1995) Grounding Language in Perception. In *Integration of Natural Language and Vision Processing (Volume I): Computational Models and Systems*, P. Mc Kevitt (Ed.), 207-227. London, England: Kluwer Academic Publishers.

Wahlster, W., N.Reithinger and A. Blocher (2001) SmartKom: towards multimodal dialogues with anthropomorphic interface agents. In *Proceedings of The International Status Conference: Lead Projects, "Human-Computer Interaction"*, G. Wolf and G. Klein (Eds.), 23-34.

Berlin, Germany: Deutsches Zentrum für Luft- und Raumfahrt.