Conventions and Notations for Knowledge Representation and Retrieval

Philippe Martin

Griffith University, School of Information Technology,
PMB 50 Gold Coast MC, QLD 9726 Australia
philippe.martin@gu.edu.au

Abstract. Much research has focused on the problem of knowledge accessibility, sharing and reuse. Specific languages (e.g. KIF, CG, RDF) and ontologies have been proposed. Common characteristics, conventions or ontological distinctions are beginning to emerge. Since knowledge providers (humans and software agents) must follow common conventions for the knowledge to be widely accessed and re-used, we propose lexical, structural, semantic and ontological conventions based on various knowledge representation projects and our own research. These are minimal conventions that can be followed by most and cover the most common knowledge representation cases. However, agreement and refinements are still required. We also show that a notation can be both readable and expressive by quickly presenting two notations – Formalized English (FE) and Frame-CG (FCG) – that we have derived from CG [9] and Frame-Logics [4]. These notations support the above conventions, and are implemented in our Web-based knowledge representation and document indexation tool, WebKB[1] [7].

1 Introduction

In [7], we argued that to permit *precise, flexible and scalable* retrieval and exploitation of knowledge representations (e.g. conceptual ontologies) and data indexed by them, the used metadata/knowledge representation languages should possess an expressive, intuitive and concise linear form; permit the indexation of any document and part of document; support the use of undeclared terms; and permit the specification of paths in a semantic network. We argued against the direct use of XML-based languages such as RDF[2], and shown how WebKB and its languages satisfy these requirements.

Precise, flexible and scalable knowledge retrieval also requires the agents (e.g. Web users or robots) that generate knowledge representations to follow conventions to permit subsequent comparison of the representations, and therefore their retrieval and merging. In this paper, we propose lexical, structural, semantic and ontological conventions based on various knowledge representation projects (especially Conceptual Graphs [9] and RDF [1]) as well as our own research [5] [6].

[1] http://meganesia.int.gu.edu.au/~phmartin/WebKB/
[2] http://www.w3.org/RDF/

We also show that a notation can be both readable and expressive by introducing two notations – Formalized English (FE) and Frame-CG (FCG) – that we derived from CG and Frame-Logics. These notations support the above conventions and associated facilities, and are implemented in our Web-based knowledge representation and document indexation tool, WebKB [7].

2 General conventions

Our conventions are general but, since we use the Conceptual Graph [CG] terminology to refer to components of knowledge representations, we assume these representations can be translated into such a directed graph model.

2.1 Lexical normalisation

InterCap style for identifiers XML[3] has become the *de facto* standard for data exchange, and RDF[4] and its XML notation ("RDF/XML") will probably become the standard for metadata exchange. Therefore, it seems important that identifiers within knowledge representations have legal XML names. This is not particularly restrictive (URLs are permitted). More importantly, the "InterCap style" has been adopted in RDF for expressing terms, with a lower case first letter for relation types[5] – as in *rhetoricalRelation* and *subClassOf* – and an upper case first letter for concept types[6] – as in *TaxiDriver*. These naming conventions have also been adopted by the "Meta Content Framework Using XML"[7].

High-level lexical facilities To be used widely and reduce lexical problems, high-level languages or query interfaces should provide lexical facilities for the user. For instance, as is the case in WebKB, language analyzers should automatically normalize identifiers that include uppercase letters, dashes or underscores into the Intercap style, as well as exploit user-defined aliases.

Such analyzers also accept queries or representations that use undeclared type names (e.g. common words) when the relevant type names can be automatically guessed via the structural and semantic constraints in the queries or representations and the ontologies they are based upon. When different interpretations are possible, the user should be alerted to make a choice. This last facility, detailed in [7], is particularly interesting when the exploited ontologies reuse a natural language lexical database such as WordNet[8]: it spares the user the complex and tedious work of declaring and organizing each term used. This facility (along with high-level notations and interfaces) seems an essential step to encourage Web (human) users to build knowledge representations. Similar ideas for the exploitation of lexical databases such as WordNet are developed in [3].

[3] http://www.w3.org/XML/
[4] http://www.w3.org/RDF/
[5] http://www.w3.org/TR/REC-rdf-syntax/#usage
[6] http://www.w3.org/TR/1998/WD-rdf-schema/#intro
[7] http://www.w3.org/TR/NOTE-MCF-XML/#secA.
[8] http://www.cogsci.princeton.edu/˜wn/

Nouns for identifiers Generally a sentence can be rephrased to avoid the use of adjectives and verbs (with the exception of "to be" and "to have"). For instance, "A cat named Tom jumps toward a wooden table" may be rephrased into "The cat which has for name Tom is agent of a jump that has for destination a table the material of which is some wood". This sentence – which is a correct sentence in Formalized English (FE) – seems unnatural but makes the concepts and their relations explicit and therefore exploitable by an automated analyzer.

The convention of using nouns, compound nouns or verb nominal forms whenever possible within representations not only makes them more explicit, it also efficiently reduces the lexical and structural ways they may be expressed. It therefore increases the possibilities of matching them.

Concept types denoted by adjectives[9] can rarely be organized by generalization relations but may be decomposed into concept types denoted by nouns. Concept types denoted by verbs can be organized by generalization relations (though the organization of the top-level types is difficult) but cannot be inserted into the hierarchy of concept types denoted by nouns (and therefore cannot be compared with them) unless verb nominal forms are used. These nominal forms, e.g. *Driving*, also recall the need to represent the time-frame or frequency of the referred processes. For similar reasons, value restrictors should also be represented via noun phrases, e.g. *ImportantWeightForAMouse* and *ImportantWeightForAnElephant*, rather than via adjectives such as *Important*.

Most identifiers in current ontologies are nouns (e.g. the Dublin Core[10] or the Upper Cyc Ontology[11]), even in relation type ontologies such as the Generalized Upper Model[12] relation hierarchy. Avoiding adverbs for relation type names is sometimes difficult, e.g. for spatial/temporal relations. However, this does not create problems in organizing relation types by generalization relations. What should be avoided is the introduction of relation type names such as *isDefinedBy* and *seeAlso*. Better names are *definition* and *additionalInformation*. These names are consistent with the usual reading conventions (e.g. in CG [9] and RDF[13]) of graph triplets {concept source, relation, concept destination}:
"`<concept source> HAS FOR <relation> <concept destination>`" or
"`<concept source> IS <relation> <concept destination>`" or
"`<concept destination> IS THE <relation> OF <concept destination>`".

Singular nouns for identifiers Most identifiers in ontologies are singular nouns. Category names must be in the singular in the Meta Content Framework Using XML. It is therefore better to avoid the introduction of plural identifiers whenever possible, e.g. by using keywords within representations such as the CG keyword `Dist` that specifies that a distributive referent is distributive.

[9] We refer to types representing the meanings of some adjectives, not misnamed types such as *Abstract* when "Abstract Entity" is actually the intended meaning
[10] http://purl.oclc.org/dc/
[11] http://www.cyc.com/cyc-2-1/cover.html
[12] http://www.darmstadt.gmd.de/publish/komet/gen-um/node11.html
[13] http://www.w3.org/TR/REC-rdf-syntax/#statement

2.2 Structural and semantic normalisation

We have seen that lexical conventions and facilities influence the structural and semantic aspects of representations. We now focus on these aspects.

First-class relations In CG and RDF, a relation is not *local* to an object, it is a first-class object itself and can be connected to any instance of the types given in the signature of the relation. This permits distributive developments (since anyone may represent anything about any object) and eases the process of comparing representations (since all terms are inter-related). This still permits the representation of relations necessarily or typically associated to objects of a particular type. Thus, though most frame-based systems also allow local relations, it is better to avoid them for the sake of knowledge reuse.

Binary basic relations Most frame-based models (including RDF) only have binary and unary relations. It is therefore better for knowledge reuse to use only unary or binary relations in languages such as CG that allow n-ary relations. Relationships of arity greater than 2 may always be represented using structured objects or collections, or more primitive binary relations. For instance, "the point A is between the points B and C" may be represented using the binary relation type *between* and a collection object grouping B and C, or using the relation types *left* and *right*, *above* and *under*, etc. Most often, decomposition makes a representation more explicit, precise and comparable with other representations.

Thus, relations should rather refer to simple/primitive relationships. As a rule of thumb, relations should not refer to processes and should rather be named with simple "relational nouns", e.g. *part* and *characteristic*. Some complex relational nouns such as *child* and *driver* are often too handy to be avoided but imply additional lexical or structural facilities (e.g. those of Ontoseek [3]).

Avoid disjunctions, negations and collections Representations that include disjunctions, negations or collections are generally less efficiently exploitable for logical inference than conjunctive existential formulas and IF-THEN rules based on these formulas [14].

It is often possible to avoid disjunctions and negations without loss of expressivity using IF-THEN rules or by exploiting type hierarchies. For instance, instead of writing that an object X is an instance of *DirectFlight* OR of *IndirectFlight*, it is better to declare X as an instance of a type *Flight* that has *DirectFlight* and *IndirectFlight* as exclusive subtypes (i.e. types that cannot have common subtypes or instances). Exclusion links between types or in some cases between whole formulas are kinds of negations that can be handled efficiently, and are included in many expressive but efficient logic models, e.g. Courteous logic on which the Business Rules Markup Language (BRML) is based.

[14] http://www.oasis-open.org/cover/brml.html

The introduction of identifiers for collections may also be avoided using keywords such as `Dist` to specify a distibutive interpretation or `Col` for a collective interpretation. Type definitions are also a way of representing facts about collections of objects that knowledge representation systems generally handle more efficiently than if collections are directly used.

Contexts are often unavoidable for expressivity sake but they can be handled efficiently if treated as positive contexts [2], that is, when only their structures are taken into account, and not the special semantics of the terms they include.

For efficiency reasons again, many knowledge representation systems, especially frame-based systems, work on rooted graphs. In a rooted graph, there is a head node representing the "central" object of the representation, the object the other nodes detail. Therefore, to help the translations of representations for various kinds of systems, it seems better to begin each representation with its central object, whatever the language used.

Precision, term definitions and constraints The more precise the representations are, the less chance they conflict with each other and the more they can be cross-checked, merged and exploited to answer queries adequately. Hence, constraints should be associated to types, and representations should rather be contextualized in space, time and author origin. No relevant concept should be implicit. For instance, instead of representing that "birds fly", it seems better to represent that "a study made by Dr Foo (Foo@bird.org) found that in 1999, 93% of healthy birds can fly" and categorize the species of birds under the exclusive subtypes *BirdWhichCanNormallyFly* and *BirdWhichCannotNormallyFly*.

Most notations make it very difficult to represent such precise statements. Even Sowa's CG linear format [9] needs to be extended to refer to the distributive interpretation of 93% of all instances of a type X; we use the form "[X: ∀ @93%]" in the CG representation of the previous example: [15]

```
[Description: [Situation: [PhysicalPossibility:
   [[Bird:λ]->(Chrc)->[Health:@good]: ∀  @93%]<-(Agent)<-[Flight]
                           ] ]->(Time)->[Date:"1999"]
]->(Source)->[Study]->(Author)->[Person: Foo@bird.org]
```

In Section 3.1 below, we give the translations of this example in FCG and FE to show that more intuitive notations are possible and useful.

Before doing so, let us note that representations which include precise domain-oriented terms should still be retrievable via queries which include more general natural language terms. A way to do this is to specialize the terms of a natural language ontology such as WordNet with the domain-oriented terms. Extending such an ontology is often quicker (and safer) than creating an ontology from scratch, ensures a better reusability of the representations and automatic comparisons with representations based on the same ontology. These issues are discussed and implemented in Ontoloom/Powerloom[16].

[15] It would have been very difficult to represent "birds of most species can fly" and the result hardly exploitable for logical inference.

[16] http://www.isi.edu/isd/OntoLoom/hpkb/OntoLoom.html#RTFToC18

3 Notations

To permit Web users precisely index Web documents, or more generally, represent knowledge, the notation(s) they use needs to be both expressive and intuitive. Otherwise, they will not use it or they will be forced or encouraged to represent information inadequately, which greatly reduces the value of the representations. A usual concern is that an increase in expressiveness leads to a model and a language too complex to handle efficiently. Actually, with structured models such as frames, RDF or CG, an inference engine may easily take more or less features into account to provide the degree of precision/efficiency required by a function or an application. For instance, a search engine can do a good and efficient job exploiting simple structure matching techniques and considering all contexts (e.g. modalities and negation) as positive contexts, so long as the retrieved representations are displayed with their associated contexts.

In this section, we list knowledge representation cases that are common in natural language sentences but rarely taken into account by current general-purpose knowledge representation languages. We do not assume any kind of exploitation or formalization. We simply show how these cases can be represented in the CG linear form (or point out the need for additional syntactic sugar), and how the FCG or FE notations are often more readable. FCG and FE are alternative notations to the CG linear form (and therefore CGIF). Because arrows have been removed and common English articles or other "modifier" words can be used as quantifiers, these notations are simpler than the CG linear form (and sometimes more expressive where set quantification and modalities are concerned). A similar "frame-like" notation using "a", "the" and "every" as quantifier Keywords is also proposed by the Knowledge Machine [8] for readability reasons (however, it does not have other quantifiers). We cannot detail these notations here but their EBNF grammars and the Yacc+Lex grammars for their translation to CG are Web-accessible [17]. All the terms in our examples – apart from keywords, URLs and some relation types – are WordNet nouns.

3.1 Presentation of FCG and FE

Here is the previous example represented in FE then in FCG.

~~~
```'93% of [bird with chrc a good health] can be agent of a flight'
  time 1999' with source a study that has for author Foo@bird.org'.
 [[[93% of(bird,chrc:a good health),agent of#:a flight],time:1999],
    source: (a study, author: Foo@bird.org)]
~~~

Contexts are delimited by square brackets in FCG and quotes in FE. At the same context level, structuration is done via parenthesis in FCG and the use of comma or keyword "and" in FE. Lambda-expressions are delimited by parenthesis in FCG, square brackets in FE. Apart from these distinctions, both notations share the same features: the quantifier keywords (e.g. "a" and "the"

[17] http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/grammars/

as existential quantifiers, "several" and "at least" as collection quantifiers), the lexical facilities (e.g. the automatic normalization of terms) and other facilities (e.g. the automatic typing of contexts – and creation of intermediary contexts if necessary – according to the signatures of the relations connected to them).

3.2 Existential statements and contexts

Most general-purpose knowledge representation languages permit the representation of existential statements and contexts, e.g. CG, KIF and RDF. Contexts are needed for referring to the source of a representation in a document and it is handy to be able to use the name (or URL) of the referred document as individual identifier. The next FCG illustrates how a representation process can itself be represented (using a concept) and how this permits to explicit details of the representation (using binary relations). It also show that an arbitrary sentence of a document (here the title) can be represented. Special delimiters – here "$(" and ")$" – are needed to encapsulate sentences in various kinds of languages.

```
[a representation, agent: philippe.martin@gu.edu.au, language: FE,
  ontology: www.int.gu.edu.au/~phmartin/WebKB/kb/KADS1ontol.html,
  creationDate: "21/01/1999", expirationDate: "22/7/9999",
  source: [http://www.int.gu.edu.au/~phmartin/WebKB/kb/KADS1.html,
             part: the title],
  object: $( KADS-I models in CG )$, /* <- text of the title */
  result: $( KADS-I has for part several models that are object of
     a representation which has for language CG. )$ /* FE repr. */
]
```

3.3 Collections and intervals

Sowa [9] uses the symbols *Dist*, *Col* and *Cum* to explicit the distributive, collective or cumulative interpretation of a collection referent. It seems clear that *Dist* or *Col* should not appear in more than one concept of a CG since this would lead to ambiguities, but the use of coreferences between collection concepts has not been explicited by Sowa. The following example illustrates the need to explicit some collection related features. ("E:" introduces the English version).

```
E:   Together, Fred, Tom and another man approved a resolution.
     A certain resolution was approved by each of them.
CG:  [Person: *s Col{Fred,Tom,*}@3]<-(approver)<-[Resolution]
     [Set:*s]   [*s Dist{*}]<-(approver)<-[Resolution: @certain]
FCG: [*g the group of persons *p {Fred,Tom,a man},
        approver of: a resolution]  [a resolution, approver: *p]
```

We used the CG `[Set:*g]` to specify that each member of the group `*g` is different. In FCG, a collection is by default a set (the distributive interpretation is the default but the collective interpretation can be specified with the keyword `group` and referred to via a variable – here `*g`). We used Sowa's keyword `@certain` to specify that each member approved the same resolution. In FCG, this is explicited via the order of the concepts (as it is in English; we have adopted this solution because it is intuitive and it also permits us to combine other kinds of quantifiers as the next examples show). The 's' at the end of the terms used for representing collections are automatically removed by the FCG parser.

Many keywords or special types need to be specified to permit the explicitation of collections, that is, (i) their kinds: `Set`, `Bag`, `OR-Bag` ("alternatives" in RDF: *rdf:Alt*), `XOR-set`, etc. (ii) their restrictors: `most`, `mostly`, `at most`, `dozens`, etc. Below is an example involving two collections ("`*r`" is supposed pre-declared). The CG statement is ambiguous (the scopes of the quantifiers are not explicit).

```
E:   At least 3 persons, including Fred,
       have each approved most of the resolutions "r".
CG:  [Person: Dist{Fred,*}@>=3]<-(Approver)<-[*r {*}@most]
FCG: [most of *r, approver: at least 3 persons {Fred,*}]
```

As shown in the next two examples, intervals can be represented (at least in FCG) using XOR-sets or normal sets.

```
E:   Tom ran between 14 min and 15 min.
FCG: [Tom, agent of: (a run, time: 14 to 15 minutes)]
E:   Tom ran between 14hrs and 15hrs (2pm to 3pm).
FCG: [Tom,agent of:(a run,time: from 14 to 15 hour__time_of_day)]
```

3.4 Universal quantifiers

As we have seen in our first example, the keywords for collections are handy to reuse for quantifying over the instances of a type. If no restrictor is used (as in `[File:∀]->(Author)->[Agent]`), relations *necessarily* connected to the instances of a type (i.e. necessary conditions) are defined. The use of restrictors such as "`most`" or percentages are a way to define "typical" relations. Number intervals and keywords such as "`at least`" and "`at most`" may be used for representing relationships of "entity-relationships" models, as in the FCG
`[any company, employee: at least 1 person]`.

Modalities and physical possibilities may be represented via contexts. For readability and normalisation reasons, we introduced special keywords in FE (`can` and `may`) and FCG (`#:` and `<=`). Thus, the following FCG represents that "any description describes something and *may* be believed by a cognitive agent":
`[any description, descr of: a thing, believer<= a cognitive_agent]`

For the same reasons, we allowed some operators (`=>, <=>, <=, =, !=, <, =<, >, >=`) to be used as relations types or to be used as aliases for other type names. Here is an example of the use of the logical operator `=>` and of the use of coreference for writing a readable second-order statement.

```
[[a relationType *r, chrc: transitive], =>
 [[a thing *x, *r: (a thing *y, *r: a thing *z)], => [*x, *r: *z]]
]
```

4 Terms and conventions for ontological cases

We now focus on how objects of certain categories can be inter-related.

4.1 Some general categories

Relatively few top-level concept types are required for the signatures of most kinds of relations useful for general knowledge representation, e.g. for the representation of natural language sentences or images from documents. These concept types and the constraints associated to them (e.g. the exclusion links between them) are useful for organizing ontologies, guiding knowledge modelling and preventing certain inconsistencies. We detailed this in [7] and introduced a top-level ontology of 150 top-level concept types and 150 basic relation types. We have used these concept types for organizing the upper levels of the WordNet ontology. In the remainder of this article, we focus on the most generic of these concept types and represent the necessary or possible relationships between their instances. In this way, we also propose a model for knowledge representation.

Before doing so, let us highlight with an example how these general types help render precise the knowledge representation process. We obtained a hierarchy of terms about computer and network technology. It appeared that the hierarchy was not a generalization hierarchy since it mixed various kinds of objects and therefore various kinds of relations. Here is an extract annotated with a general category for each term.

```
Computing  //process or domain?
  Computer_hardware  //physical object!
    Compiler  //hardware or software?
  Computer_software  //description!
    Software_language  //description medium!
    Applications  //process description (may or may not be software)!
Networking  //process or domain?
```

Here is, represented in FCG, a part of the ontology that we have built to make explicit the general category of each of the above terms and thus permit semantic checks and the use of relations associated to the general categories. The author of the previous hierarchy had to be contacted to determine what some of the terms meant. Indentation is only for presentation purposes. [18]

```
[Domain_object, subtype: {Computing_object Networking_object}]
  [Networking_object,partition:{{Network_software,Network_hardware}}]
[Physical_entity, subtype: Hardware]
  [Hardware, subtype: {Computer_hardware, Network_hardware}]
    [Network_hardware, partition: {{Local area network, Switch}}]
[Description, subtype: {Software, Application}]
[Descr_medium, subtype: {Software_language, Network_language}]
```

[18] The relation *partition* connects a type to a set of partitions, each being a set of exclusive types. Thus, two pairs of brackets are required even for a single partition.

Here are the generalization and exclusion relations between our most generic concept types. Indentation is only for presentation.

```
[Thing (^the supertype of all first order concept types^),
   partition: { {Situation, Entity}, {Thing_playing_a_role} }]

  [Situation (^a thing that occurs in a region of time and space^),
     partition: {{State, Process}, {Phenomenon},
                 {Situation_playing_a_role}}]

  [Entity (^a thing that may be involved in a situation^),
     partition:{{Information_entity,Temporal_entity,Spatial_entity},
                {Collection}, {Entity_playing_a_role} }]

    [Spatial_entity (^an entity that occupies a space region^),
       partition: {{Space_location, Physical_entity,
                    Imaginary_spatial_entity}}]

      [Physical_entity (^a spatial entity made of matter^),
         partition: {{Inanimate_object, Living_thing},
                     {Goal_directed_entity(^cognitive entity^)}}]

    [Information_entity (^for information or its representations^),
       partition: { {Description, Descr_container,
                     Characteristic, Measure, Measure_unit} }]

      [Description (^description of a situation^),
         partition: { {Description_content, Description_medium} }]

        [Description_content,partition:{{Belief,Hypothesis,
                                          Narration,Argument}}]

        [Description_medium,partition:{{Symbol,Syntax,Language,
                                         Abstract_data_type,Script}}]

      [Description_container,partition:{{File,Image,Hologram,
                                          Document_element}}]

      [Characteristic (^ a dimension of something ^),
         partition: {{Psychological_characteristic,
              Physical_characteristic, Situation_characteristic}}]

      [Measure, partition: {{Psychological_measure,
                             Physical_measure,Situation_measure}}]

      [Measure_unit, partition: {{Psychological_measure_unit,
                   Physical_measure_unit,Situation_measure_unit}}]

  [Thing_playing_a_role (^subcategorisation is domain-dependent^)
     partition: {{Domain_object},{Thing_needed_for_a_process},
                 {Entity_playing_a_role,Situation_playing_a_role}}]
```

4.2 Descriptions

We now focus on necessary and possible relationships from concepts of the types listed above. We begin with "descriptions" and related objects.

```
[any thing,
  descr   <= a description,           //anything may be described
  descr_in<= a descr_container,       //e.g. in a file, a hologram
]
[any description,                     //or "proposition"
  descr of       : a thing,
  descr_medium   : a descr_medium,    //symbols or a language
  descr_container: a descr_container, //e.g. in a file
  author         : 1 entity,          //unique author
  believer   <= a cognitive_agent, modality <= a modality,
  logical_relation<=a description, rhetorical_relation<=a description
]
[any descr_container, descr_support: a physical_entity,
                      descr_in of: a thing]
[any descr_medium, descr_medium of: a thing]
[ [a thing *t, descr_in: a descr_container *c],
  <=> [*t, descr: (a description, descr_container: *c)] ]
```

Examples of logical relation types are *Or* and *Xor*. Examples of rhetorical relation types are *summary*, *motivation* and *antithesis*.

4.3 Characteristics and measures

When representing characteristics, the characteristic itself should be distinguished from its measure(s), as in `[a pen, physChrc: (a length, measure: 12 cm)]`. Since this habit does not come naturally, abbreviations such as the following should probably be adopted as conventions: `[a pen, length: 12 cm]`. However, this implies additional constraints on the ontology and on its exploitation by the analyzers, e.g. *Length* should be declared as a subtype of a type *Characteristic* and this would have to be a predefined type in any analyzer. Here are relations for explicit representations.

```
[any thing, chrc <= a characteristic]  //e.g. speed, ingenuity
[any characteristic, measure <= a measure, chrc of <= a thing]
[any measure, quantity: a number, unit: a measure_unit,
              measure of: a characteristic]
[any physical_entity, physChrc <= a physical_characteristic]
[any goal_directed_entity,physChrc<=a psychological_characteristic]
```

A lot of concept types may be found in WordNet for physical or psychological features, e.g. *Memory* and *Cognition*, but unfortunately those types are not well organized and often mixed with misclassified types such as *Mind*, *Lexicon* and *Structure*.

4.4 Situations, processes and temporal entities

Here are some representations of relationships that are – or may be – connected to situations, processes and temporal entities. Most of the relations types were proposed by [9].

```
[any situation,
   s_succ of: a situation,  s_succ: a situation,
   location : a spatial_entity,
   time: a temporal_entity, duration : a duration,
   situationChrc<= a situation_characteristic
]
[any temporal_entity,
   time of <= a situation,  duration of <= a situation,
   temporal_order_relation <= a temporal_entity
]
[any process,
   triggering_event<= an event,       ending_event  <= an event,
   ending          <= a state,        ending of     <= a state,
   precondition    <= a state,        postcondition <= a state,
   initiator  <= a goal_directed_agent,  agent      <= an entity,
   instrument <= an entity,              object     <= a thing,
   experiencer<= a conscious_agent,      recipient  <= an agent,
   result     <= a thing,                sub_process<= a process,
   manner     <= a situation_characteristic,
   method     <= a description,       source <= a spatial_entity,
   destination<= a spatial_entity,    path   <= a spatial_entity
]
```

The relation types *input* and *output* may respectively be declared as subtypes of *object* and *result*. Further specializations are *input_output*, *object_to_modify* and *object_to_mute*.

Given we may use intervals and exclusive sets for representing time concepts, only two relation types seem necessary between situations and temporal entities: *time* and *duration*. Here are examples of two typical cases.

```
E:   On the 21/12/1999, John went to his office between 13h and 14h.
FCG: [[a travel, agent: John, destination: (an office, owner: John),
                 time: 13 to 14 hour], time: "21/12/1999"]
E:   John usually takes 20 min or 40 min to go to his office.
FCG: [most of (travel, agent: John,
                       destination: (an office, owner: John)),
         duration: {20 min | 40 min}]
```

These examples do not violate the signatures of *time* and *duration*. The following representation, which uses a context, should be considered by the analyzers as equivalent to the first of the above two examples.

```
[ [John, agent of: travel, destination: (an office, owner: John)],
         time: 13 to 14 hour]
```

4.5 Miscellaneous

Here are additional common kinds of relationships.

```
[any thing, part <= a thing] //anything may have at least 1 part
[any physical_entity, material: a physical_entity]
[any collection, subset <= a collection,
                 element<= a thing, count: a natural]

[Order_relation, domain: Thing, range: Thing,
   partition: { {Spatial_order_relation, Temporal_order_relation},
                {Meet,In,Near,Before,After} }]
[Spatial_order_relation, subtype: {On,Above,Below}]
[On, subtype of: {Meet,Above}]
```

We have defined relation types such as *meet* and *near* as direct subtypes of *order_relation* and allowed them to connect any pair of concepts. Specializations of these types, e.g. *spatial_meet* and *temporal_meet*, could be defined to allow the use of more precise and constrained types upon which further semantic checks could be done. Such precise modelling may be found in the CYC and Ontolingua top-level ontologies (for instance, 2D and 3D spatial relations are distinguished). However, we cannot expect the average user to spend his time looking for the most specific terms in such libraries. Nonetheless, these libraries could be exploited by authoring tools to automatically find more specific relations that do not violate the signatures of the general relation used.

Though relations such as *part* or *subset* are partial order relations like *subtype*, for the sake of precision, they should not be directly connected to concept types. For instance, `[Airplane, part: Wing]` might be intended to represent the fact that "any airplane has for part a wing", but many alternative interpretations are possible: "any wing is part of a plane", "a wing is part of any plane", etc.

5 Conclusion

Information can be represented in many ways. For knowledge representations to be automatically comparable, conventions must be followed by authoring agents. We have proposed general lexical, structural and semantic conventions, then examined some knowledge representations cases that are common in natural language sentences but rarely taken into account by current general-purpose knowledge representation languages. We have introduced two notations (Frame-CG and Formalized English) that support and guide the use of these conventions (e.g. the syntax, the quantifiers and restrictors - "a", "the", "several", etc. - lead to the use of nouns as identifiers), cover the listed knowledge representations cases and remain intuitive. We argued that an inference engine can exploit expressive languages efficiently (at the expense of precision) by ignoring some of the more complex features. Finally, we detailed some top-level concept types and their relationships to guide and provide semantic constraints for knowledge representation.

As highlighted above, the precise models that are found in CYC and Ontolingua top-level ontologies are certainly useful but will doubtfully be used directly by human agents to represent information or quickly index documents, sentences or images. Instead, we expect people (e.g. Web users) to utilize a small set of relation types and simply use common words for concepts types: given the signatures of the relation types, a lexical database such as WordNet may be exploited by an authoring tool to derive the relevant concept types or ask the user for more precision [3] [7]. Users will also probably use scalable multi-user knowledge servers to refer, fix and complement lexical databases.

Acknowledgments

This work is supported by a research grant from the Australian Defense, Science and Technology Organisation. We also thank the reviewer who pointed out that the Knowledge Machine project [8] has similar principles to ours (frame-like notation with English articles, type partitions, etc.) and proposes a situation-based inference engine.

References

1. Berners-Lee, T.: The Semantic Toolbox: Building Semantics on top of XML-RDF. http://www.w3.org/DesignIssues/Toolbox.html (W3C Note, 24 May 1999); see also the "Semantic Web Road map" at http://www.w3.org/DesignIssues/Semantic.html, and the "Web Architecture: Describing and Exchanging Data" at http://www.w3.org/1999/04/WebData
2. Chein, M., Mugnier, M.L.: Positive Nested Conceptual Graphs. In: ICCS'97, 5th International Conference on Conceptual Structures, Springer Verlag, LNAI 1257 (1997) 95–109
3. Guarino, N., Masolo, C., Vetere, G.: Ontoseek: Content-based Access to the Web. In: IEEE Intelligent Systems, Vol. 14, No. 3 (1999) 70–80
4. Kifer, M., Lausen, G., Wu, J.: Logical Foundations of Object-Oriented and Frame-Based Languages. In: Journal of the ACM, vol 42, 1995.
5. Martin, Ph.: Using the WordNet Concept Catalog and a Relation Hierarchy for Knowledge Acquisition. In: Peirce'95, 4th Peirce workshop, California (1995) http://www.inria.fr/acacia/Publications/1995/peirce95phm.ps.Z
6. Martin, Ph.: *Exploitation de graphes conceptuels et de documents structurés et hypertextes pour l'acquisition de connaissances et la recherche d'informations*, PhD Thesis, University of Nice - Sophia Antipolis, France (1996)
7. Martin, Ph., Eklund, P.: Embedding Knowledge in Web Documents: CGs versus XML-based Metadata Languages. In: ICCS'99, 7th International Conference on Conceptual Structures, Springer Verlag, LNAI 1640 (1999) 230–246. URL http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/papers/iccs99/iccs99.ps
8. Clark, P., Porter, B.: KM: The Knowledge Machine. http://www.cs.utexas.edu/users/mfkb/km.html
9. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading, MA (1984)
See also: http://www.bestweb.net/~sowa/ontology/index.htm