

A (Very) Short Introduction to Description Logics

The following examples illustrate how to use a description logic to represent and reason about types and objects, in particular with the LOOM knowledge representation system.

- *Letters are either vowels or consonants. The vowels are a e i o u y.*

```
(defconcept letter
  :exhaustive-partition (vowel consonant))

(defconcept consonant
  :is letter)

(defconcept vowel
  :is (and letter
      (one-of a e i o u y)))

(tellm (vowel a) (vowel e) (vowel i) (vowel o) (vowel u))
(tellm (letter y) (letter p) (letter n))
```

Each individual vowel is defined as an instance. Notice that we defined *y* as an instance of letter. LOOM uses the definition of vowel and deduces that *y* is a vowel but *p* and *n* are not. It also deduces that *a e i o u* are all letters as well as vowels.

Vowels and consonants form what is called an *exhaustive partition* of letter. This means that any letter has to be either a vowel or a consonant, and that no letter can be both.

- *One or more letters form a word. A string can contain letters.*

```
(defconcept string
  :is-primitive (at-least 1 has-element))

(defconcept word
  :is (and string
      (all has-letter letter)
      (at-least 1 has-letter)))
```

We can specify additional information about any concept at any time. For example, we may add:

```
(defconcept string
  :constraints (at-least 0 has-letter))
```

Notice that since some strings are a mix of letters and numbers, having some letter is not part of the definition of *string* but instead is a constraint. However, it is part of the definition of *word*.

```
(defrelation has-element :domain string)
(defrelation has-letter :is (and has-element :range letter))
```

The domain of a relation determines the class of objects for which it can be defined. The range determines the class of objects that can be given as a value (a filler) of the relation. Relations can be defined in terms of other relations, like `has-letter`.

- *Vowel words are a class of words whose letters are vowels. 3-letter words are a class of words that have 3 letters or less.*

```
(defconcept vowel-word :is (:and word (all has-letter vowel)))

(defconcept 3-letter-word :is (:and word (at-most 3 has-letter)))
```

If we now define an instance:

```
(tellm (:about Spain (has-letter a) (has-letter i) (has-letter p) (has-letter n)))
```

Based on the definitions, LOOM will deduce that `Spain` is not a vowel-word and it is not a 3-letter word. It will also deduce that `Spain` is a string. LOOM will not deduce that `Spain` is a word, because it is possible that we specify later that it has an element that is a number.¹

- *Some words are country names.*

```
(defconcept country-name :is-primitive word)
```

This concept is defined as primitive, which means that the system will not be given a way to determine when a word is a country name. The user will indicate explicitly which words are country names. Thus, `Spain` is not considered a country name until we specify so:

```
(tellm (country-name Spain))
```

The following table shows some of the constructors that can be used in LOOM definitions. C and D represent concepts, R represents relations, and k represents a number.

Constructor	Meaning
<code>(and C D)</code>	conjunction
<code>(or C D)</code>	disjunction
<code>(all R C)</code>	all the values of R are C
<code>(at-least k R)</code>	R has at least k values
<code>(at-most k R)</code>	R has at most k values
<code>(exactly k R)</code>	<code>(at-least k R)</code> and <code>(at-most k R)</code>
<code>(some R C)</code>	<code>(at-least 1 R)</code>
<code>(the R C)</code>	<code>(all R C)</code> and <code>(exactly 1 R)</code>

¹For example, we still need to specify `(tellm (:about Spain (has-letter s)))`. In any case, LOOM is able to deduce that `Spain` is a word once we indicate that we are done with our definitions with `(set-features :closed-world)`. See the LOOM manual for more details.