6.1 Simple Expression Constraints

The simplest type of expression constraint contains a single concept optionally preceded by an expression constraint operator and/or membership function. Expression constraint operators (e.g. descendant of) traverse the hierarchical relationships in SNOMED CT to return the set of concepts that are directly or transitively connected to the focus concept. Membership functions return the set of concepts referenced by a reference set. In this section we consider some of these simple examples.

Self

If no expression constraint operator or membership function is applied, the expression constraint is satisfied only by the specified concept. For example, the expression constraint below is satisfied only by the concept 404684003 |Clinical finding|.

```
404684003 |Clinical finding|
```

Please note that this expression constraint is equivalent to an expression that looks the same but is written in SNOMED CT Compositional Grammar.

Descendant of

A single 'less than' sign (i.e. "<") indicates that the expression constraint is satisfied by all descendants of the specified concept. The expression constraint below evaluates to the set of all subtypes (both direct children and transitive subtypes) of 404684003 |Clinical finding|, using the brief syntax.

```
< 404684003 |Clinical finding|
```

Using the long syntax, the above expression constraint may be represented as:

```
descendantOf 404684003 |Clinical finding|
```

The descendantOf function is primarily used on concepts, which serve as the ‘grouper’ of a set of values (e.g. |Clinical finding|, |Clinical finding (finding)|, |Severities (qualifier value)|, |Unit (qualifier value)|). The descendantOf function may also be applied to other concepts, or to nested expression constraints (as discussed in 6.7 Nested Expression Constraints).

Descendant or Self of

Two consecutive 'less than' signs (i.e. "<<") indicates that the expression constraint is satisfied by all descendants of the specified concept plus the specified concept itself. The expression constraint below evaluates to the set of descendants of 73211009 |Diabetes mellitus|, plus the concept 73211009 |Diabetes mellitus| itself.

```
<< 73211009 |Diabetes mellitus|
```

Using the long syntax, the above expression constraint may be represented as:

```
descendantOrSelfOf 73211009 |Diabetes mellitus|
```

The descendantOrSelfOf function is primarily used for attribute values, which refer to a specific clinical value (e.g. 73211009 |Diabetes mellitus|, 73761001 |Colonoscopy|, 385055001 |Tablet dose form|), but any specialization of this value is also acceptable. The descendantOrSelfOf function may also be applied to other concepts, or to nested expression constraints (as discussed in 6.7 Nested Expression Constraints).

Child of

A 'less than' sign directly followed by an exclamation mark (i.e. "!<") indicates that the expression constraint is satisfied by the set of proximal children of the specified concept. The children of a concept are those concepts that are the source of a non-redundant relationship whose target is the given concept. The expression constraint below, represented using the brief syntax, evaluates to the set of immediate children of the concept 404684003 |Clinical finding|.

```
!< 404684003 |Clinical finding|
```

Using the long syntax, the above expression constraint may be represented as:

```
childOf 404684003 |Clinical finding|
```
Please note that the childOf function may only be executed against a finite and pre-classified substrate, and that the results of this function are specific to the substrate used. The childOf function may also be applied to nested expression constraints (as discussed in 6.7 Nested Expression Constraints).

### Ancestor of

A single ‘greater than’ sign (i.e. ">") indicates that the expression constraint is satisfied by all ancestors of the specified concept. The expression constraint below, using the brief syntax, evaluates to the set of all supertypes (both direct parents and transitive supertypes) of 40541001 | Acute pulmonary edema|:

```plaintext
> 40541001 | Acute pulmonary edema|
```

Using the long syntax, the above expression constraint may be represented as:

```plaintext
ancestorOf 40541001 | Acute pulmonary edema|
```

Please note that the ancestorOf function may also be applied to nested expression constraints (as discussed in 6.7 Nested Expression Constraints).

### Ancestor or Self of

Two consecutive ‘greater than’ signs (i.e. ">>") indicates that the expression constraint is satisfied by all ancestors of the specified concept plus the specified concept itself. The expression constraint below evaluates to the set of ancestors of 40541001 | Acute pulmonary edema|, plus the concept 40541001 | Acute pulmonary edema|:

```plaintext
>> 40541001 | Acute pulmonary edema|
```

Using the long syntax, the above expression constraint may be represented as:

```plaintext
ancestorOrSelfOf 40541001 | Acute pulmonary edema|
```

Please note that the ancestorOrSelfOf function may also be applied to nested expression constraints (as discussed in 6.7 Nested Expression Constraints).

### Parent of

A ‘greater than’ sign directly followed by an exclamation mark (i.e. "%!") indicates that the expression constraint is satisfied by the set of proximal parents of the specified concept. The parents of a concept are those concepts that are the target of a non-redundant relationship whose source is the given concept. The expression constraint below, represented using the brief syntax, evaluates to the set of immediate parents of the concept 40541001 | Acute pulmonary edema|:

```plaintext
>! 40541001 | Acute pulmonary edema|
```

Using the long syntax, the above expression constraint may be represented as:

```plaintext
parentOf 40541001 | Acute pulmonary edema|
```

Please note that the parentOf function should only be executed against a finite and pre-classified substrate, and that the results of this function are specific to the substrate used. The parentOf function may also be applied to nested expression constraints (as discussed in 6.7 Nested Expression Constraints).

### Member of

The memberOf function evaluates to the set of concepts that are referenced by the given reference set (i.e. the set of referencedComponentIds). Please note that this function may be applied only to reference sets whose referenced components are concepts. The SNOMED CT Expression Constraint Language does not support use of the memberOf function on reference sets whose referencedComponents are not concepts (i.e. descriptions or relationships).

The memberOf function is represented in the brief syntax using a ‘caret’ character (i.e. "^") and is usually followed by a single concept id for a concept-based reference set. For example, the following expression constraint is satisfied by the set of concepts which are members of 700043003 | Example problem list concepts reference set|:

```plaintext
^ 700043003 | Example problem list concepts reference set|
```
Using the long syntax the expression constraint is represented as:

```
memberOf 700043003 [Example problem list concepts reference set]
```

Please note that it is also possible to apply the memberOf function to an expression constraint that returns a set of concept-based reference set concepts. For more information, please refer to 6.7 Nested Expression Constraints.

Any

A single 'star' (i.e. "**") may be used in the place of a concept reference to represent any concept in the substrate. The expression constraint below evaluates to the set of all concepts in the given substrate.

```
*
```

Using the long syntax, the above expression constraint may also be represented as:

```
ANY
```

This wildcard character (or 'ANY' keyword) may be used anywhere within an expression constraint that a concept reference may be used. In many situations, the wildcard is equivalent to the following expression constraint:

```
<< 138875005 [SNOMED CT concept]
```

However, some situations exist in which the concept 138875005 [SNOMED CT concept] is not included in the substrate, and therefore cannot be used to determine the full set of concepts available. In other cases, the single character wildcard may serve as a convenient shortcut for the longer expression constraint above.

Please note that the following three expression constraints evaluate to the same set of concepts:

```
*

<< *

>> *
```

The two expression constraints below evaluate to all concepts in the substrate minus the root concept:

```
< *

<! *
```

And the two expression constraints below evaluate to all non-leaf concepts in the substrate:

```
> *

>! *
```

Finally, the expression constraint below evaluates to all concepts that are referenced by any reference set in the substrate:

```
^ *
```