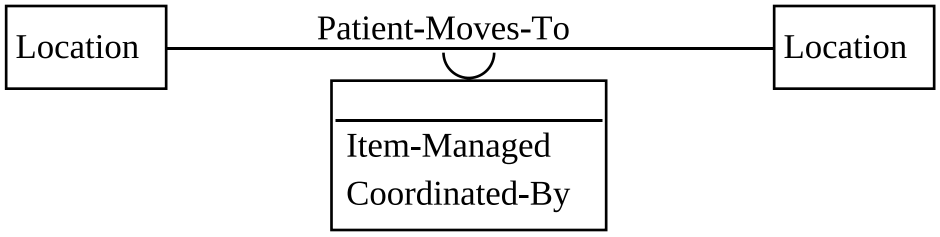
# Reification Support for Loom

## Problem Example

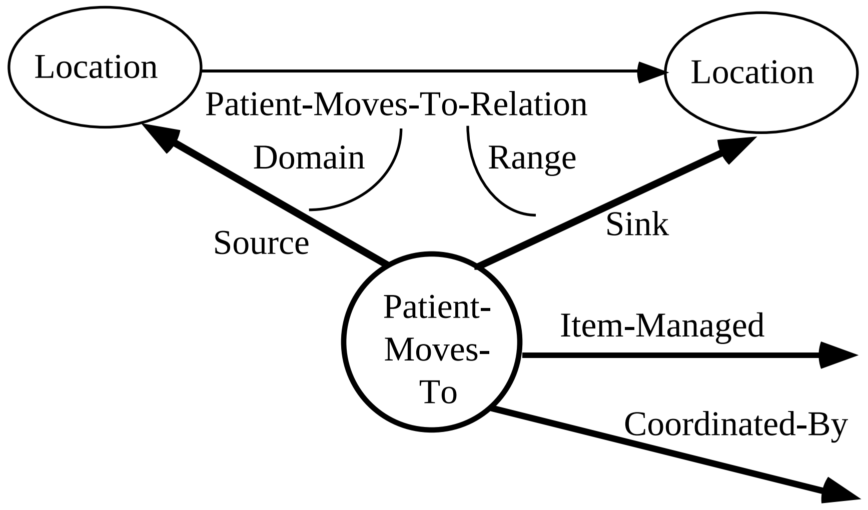
There is a need to support Rumbaugh-style link attributes for the TCIMS project. In order to make this work as seamlessly as possible, there will need to be some extensions to the Loom language. The example that we plan to use is based on the **producer-consumer** dependency and its specific example of **patient-moves-to**.



## Conceptual Overview

In order to talk about a particular relation, it is necessary in Loom to provide a reification of that relation. Unfortunately, the existing reification mechanism does not do the job properly. It is retained because of the need to support existing Loom-based applications, most notably Penman.

A proper reification would support attaching case roles to the reified relation and allow the mapping of one of those case roles to the domain of the relation and another one to the range of the relation. The case roles that are linked to the domain and range of the relation must be single-valued, since a particular instance of a relation can only link one object to one other object. Other case roles will not need to be single-valued.



## Definition Support

We will augment the **defconcept** syntax to support proper creation of a case frame that reifies a given relation. This will involve the addition of the keyword **:reifies**, which will be used to introduce the name of the reified relation (which can be the same as the conceptName) and define the link between case roles of the and the domain and range of the relation. After these required arguments, one can optionally specify additional arguments to be used in the definition of the reified relation. They can be any syntax acceptable to **defrelation** such as **:domain**, **:characteristics**, etc. (Loom will automatically setup the relation hierarchy to mirror on the concept hierarchy, so it will not be necessary to explicitly specify the mirrored relations via an **:is** clause.)

```
(defconcept conceptName …
  :reifies (relationName
            [:case-roles (domainRoleName rangeRoleName)]
            [defrelationKeyword Value] …))
```

The *domainRoleName* and *rangeRoleName* are the names of relations which will be roles of *conceptName*. They can be specified explicitly via the :case-roles keyword inside the relation definition. Otherwise default values of `source-case` or `target-case` will be used. A concrete example would be the following:

```
(defconcept producer-consumer-dependency
   :is-primitive dependency
   :constraints (:and (:all dependency-source activity)
                      (:all dependency-sink   activity)
                      (:all item-managed      object))
   :reifies (producer-consumer-relation
                 :case-roles (dependency-source
                              dependency-sink)))

(defoncept patient-moves-to
   :is-primitive (:and producer-consumer-dependency
                       (:all dependency-source location)
                       (:all dependency-sink  location))
   :constraints (:some item-managed casualty)
   :reifies (patient-moves-to-relation
                 :case-roles (dependency-source
                              dependency-sink)))
```

The definition for **patient-moves-to** is used to create a concept named “patient-moves-to” as well as a relation named “patient-moves-to-relation.” The role name arguments are used to indicate that the relation **patient-moves-to-relation** is used to link the **dependency-source** of the concept to the **dependency-sink** of the concept. The domain and range of **patient-moves-to-relation** will be set to **location** based on the restrictions in the definition of **patient-moves-to**.

In addition, the constraint that there is at least one **item-managed** of type **patient** is specified. Loom will also arrange for **patient-moves-to-relation** to be a sub-relation of **producer-consumer-relation**.

## Assertion Support

We propose to extend the assertion language through the introduction of a 3-ary function **link** which maps a relation and two instances to an instance of the reification of that relation. Assertions can then be made about this instance in the same way as any other Loom instance. These assertions describe the particular link. For example:

```
(tell (location l1) (location l2) (patient p3)
      (patient-moves-to-relation l1 l2))
(tell (item-managed
              (link patient-moves-to-relation l1 l2)
              p3))
```

These forms assert that patient p3 is the item managed by the **patient-moves-to-relation** link between the locations l1 and l2.

## Availability

Patches to add this functionality to Loom 2.1 are currently available on an experimental basis. The basic functionality exists, but there are still a few loose ends — mainly ensuring that revisions to the definitions are tracked properly and that retractions of relation fillers cause a retraction of the