

# Connector Labels

Leon Starr  
Tue Apr 08 2008

FLATLAND

Labels are text areas associated with Connectors. They can name and annotate relationships such as transitions, dependencies, messages and so forth. A set of Label Specifications may be created for each Connector Type to predefine the legal and default placements and properties of Labels filled and moved around during an edit session. Here we explore some properties of Labels, focusing especially on specifications.

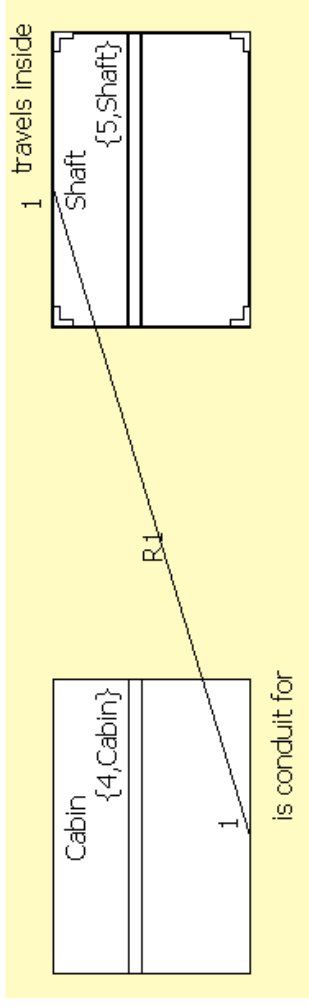
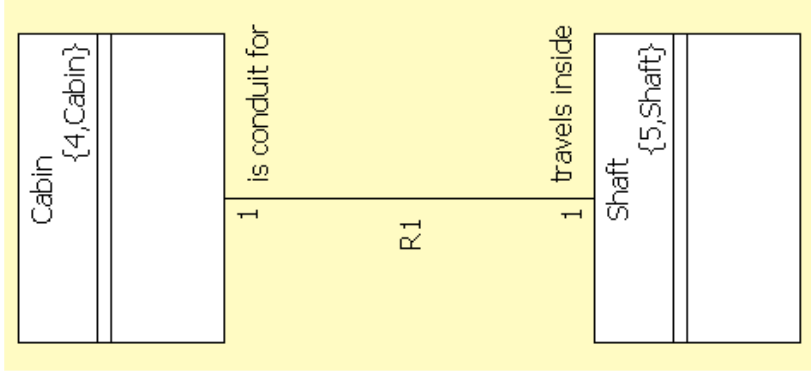
mint.flatland.tn.19 / Version 0.4



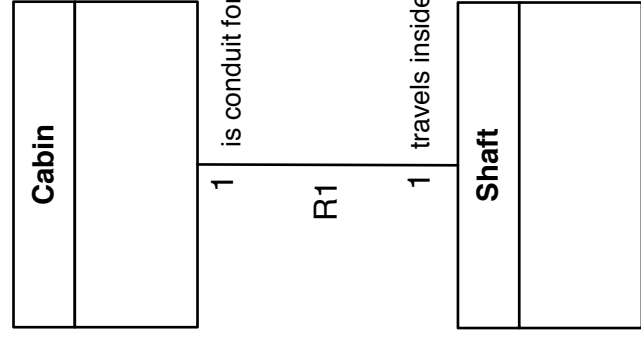
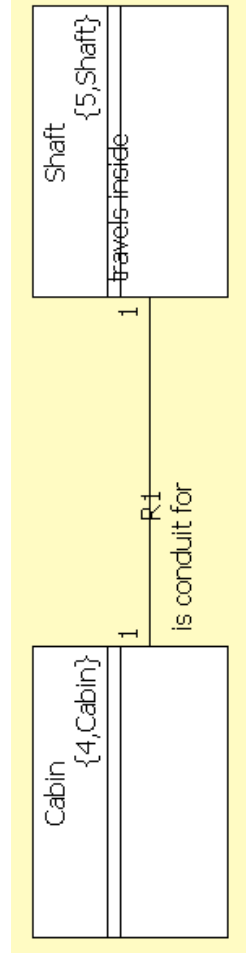
Copyright 2008, All rights reserved.  
MODEL INTEGRATION, LLC



## Examples of label misplacement after CCW rotation of a binary connector



Executable  
UML Tool



Mac  
Diagramming  
Tool



Connector rotations are not supported by either of these example tools. The Executable UML tool requires 2 steps to get the connector lined up, while the more general draw tool takes only 1. In either case, not all of the labels are lined up with regard to both the nodes and connectors. Also, the labels end up overlapping nodes, which, in a UML tool, should never happen. These problems exist to varying degrees with all UML tools.

## Label Placements and Hooks

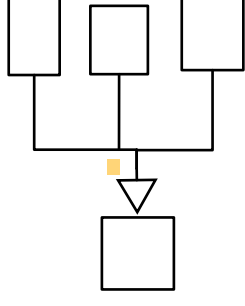
Configurations of legal label Placements\* may be defined for each Connector Type. The available Placements for a UML statechart transition will be completely different than those for a UML class model generalization or binary association. The hopping order for alternate locations may be different for each label, even on the same Connector Type.

The geometry varies widely for each of these Connector Types (reflexive, binary and group). Each has different features that may be used as reference points for label positions. Connector Types, geometry and behavior is already discussed in tn.20. Here we will focus on how positions of labels are specified and how labels can be manipulated for the various Connector Types.

A Connector Type may contain one or more Straight or Flexible Branches. Every Branch has a From and To Terminator and a computed midpoint. In addition, a Flexible Branch may have one or more Bends. We abstract the idea of a "Hook" from which we can hang one or more Labels. So we have the Terminator, Middle and Bend types of Hook. Each label Placement will be specified relative to one of these Hook types.

\*An attempt is made in all of my tech notes to capitalize the first letter of any terms corresponding to class names in the derived UML model (see attached).

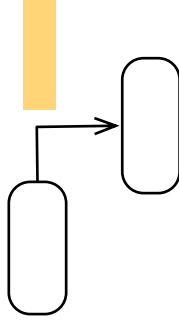
Here are some examples of different types of Connectors with Labels positioned relative to different Hooks.



**Straight Branch in a Group Connector with a label placement relative to the Middle Hook.**



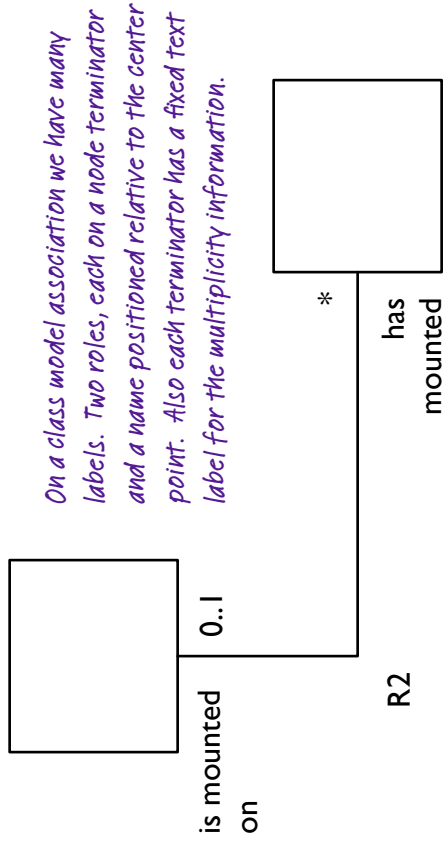
**Flexible Branches in two different Binary Connectors with Placements relative to Terminator, Middle and Bend Hooks.**



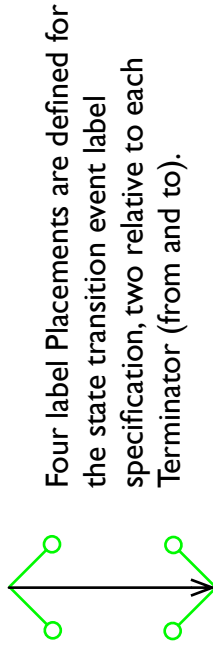
The parameters necessary to specify label configurations relative to the various Hook Types is worked out on the following pages.

## Binary Connector Labels

One or more Labels may be attached and positioned relative to either end and relative to the mid point or a bend corner point. Some Labels are open text while others are limited to a set of predefined strings. A maximum size in characters may be defined for open text.

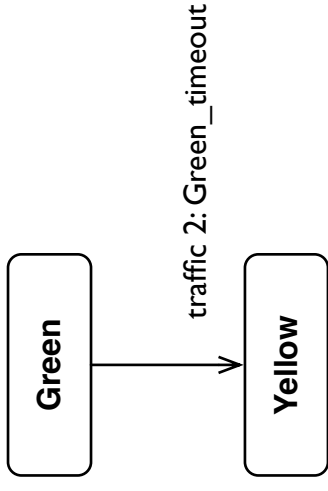


A Label may have one or more alternate locations where it can appear. This makes it easier to adapt to obstacles on a diagram. Consider the event label on a UML statechart transition.

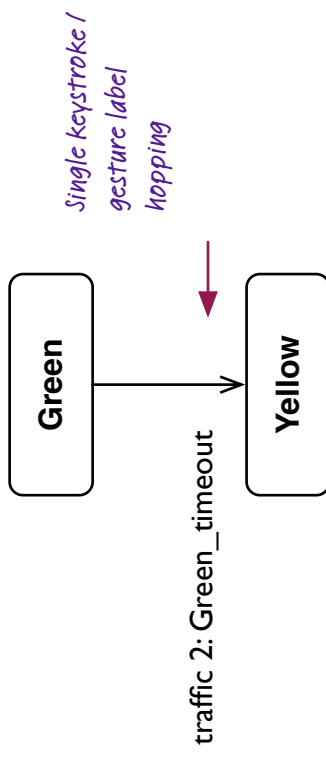


Predefined placements

A diagram using the lower right label Placement appears below.



During editing the user can easily shift the label text to one of the other three positions using single keystrokes instead of having to drag and reposition.

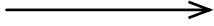


The left/right text alignment varies depending on location. In each case, the label text is aligned toward the Connector. On a horizontal layout the text will be aligned to the Node.

## Terminator Hooks

There is a Terminator Hook on each end of a Straight or Flexible Branch. A Hook is an invisible reference point where we can "hang" one or more Labels.

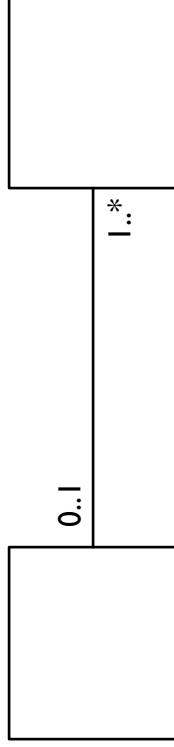
### Hook on From Terminator



*Note that a Binary Connector consists only of a single Branch so it is easy to confuse the terms "Branch" and "Binary Connector". Remember that a Branch is a component of a Connector. In this case, the only one.*

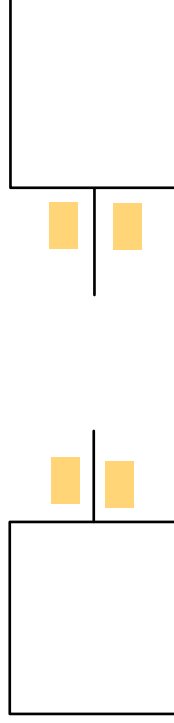
### Hook on To Terminator

Using the UML class diagram binary association as an example, we will define Placements for the multiplicity Labels.



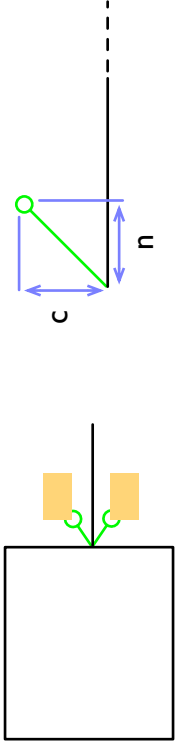
*Strictly speaking, there is no directionality on a UML class binary association. We could have called them A/B terminators instead, but, for consistency, we will always use to/from to designate each end of the Branch (in a Binary Connector).*

In this example we need to specify the locations of two separate Labels. There is the "from" multiplicity and the "to" multiplicity Label. Each Label has two alternate locations as shown.

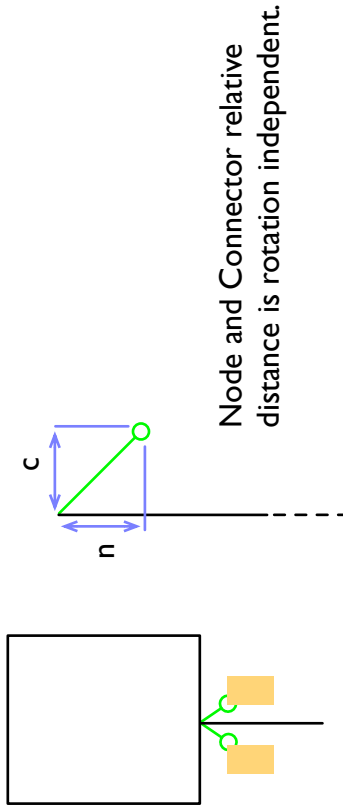


**Important:** This means that if you are positioning the "from" multiplicity Label, you can only hop among locations on the "from" side. This is in marked contrast to the state transition event example coming up.

## Terminator Hooks - UML multiplicity example



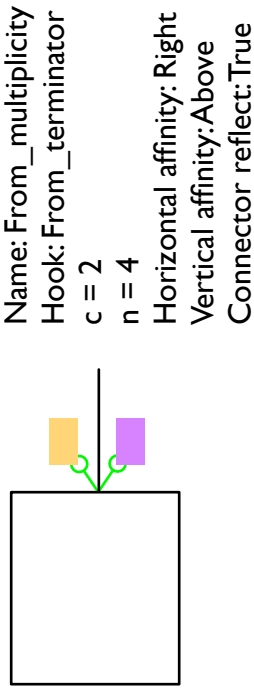
There is a distance from the Node ( $n$ ) and a distance from the Connector ( $c$ ). To avoid overlap with the node or connector, we will require these to be positive distances.



Node and Connector relative distance is rotation independent.

Right, but how do we know on which side of the Connector to position the Label? Sometimes both sides may be okay, other times just one side is acceptable. Furthermore, the sides may be right/left or above/below depending on the Connector orientation. So we can define horizontal and vertical affinities along with some indication as to whether the opposite side is an acceptable alternate Placement. Here is an example of a complete, rotation independent Terminator Hook referenced Placement specification.

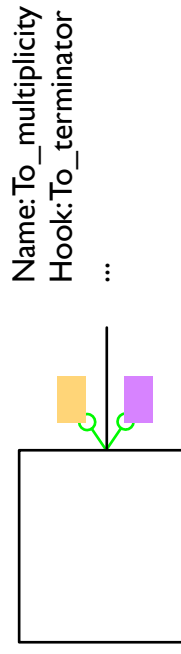
## Multiplicity label Placement values



Name: From\_multiplicity  
 Hook: From\_terminator  
 $c = 2$   
 $n = 4$   
 Horizontal affinity: Right  
 Vertical affinity: Above  
 Connector reflect: True

This means that the multiplicity label will normally appear above or to the right of the Connector Hook and 2 units away from the node side of the Terminator Hook and 2 units away from the Connector. The same location on the other side of the Connector will be available as an alternate location.

The other multiplicity Label (To\_multiplicity) is the same except with the following change:

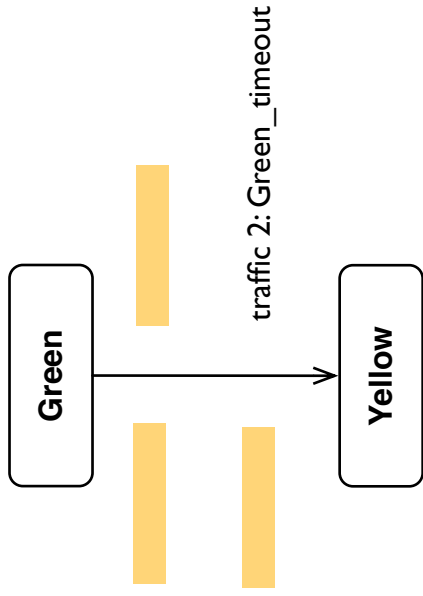


Name: To\_multiplicity  
 Hook: To\_terminator  
 ...

Now lets consider a similar, but slightly different case - the UML statechart transition.

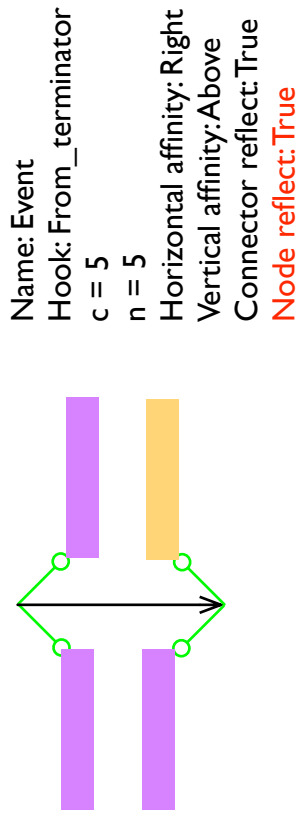
## Terminator Hooks - UML transition event example

The Executable UML state transition is directional and has only a single Label - the event. There are many legal positions for such a Label.



Even though the previous UML class association example had similar symmetry, the `Node_reflect` would have to be set to `False` in that case. This is because there are two distinct multiplicity Labels. Here, there is only a single event Label with one primary and three alternate locations.

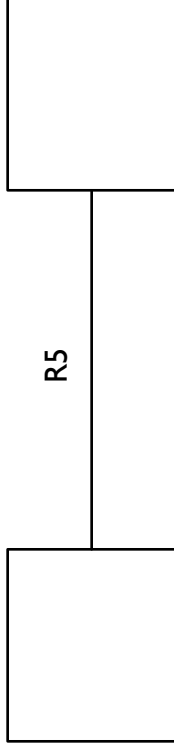
We can define the placement configuration shown above with a single Placement specification as shown:



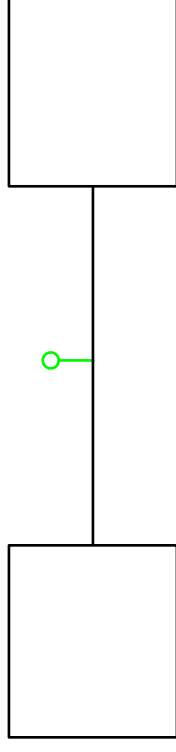
Only the `Node_reflect` parameter has been added to our Placement specification. This saves us the trouble of having to create a nearly identical Placement spec for the `To_terminator` Hook.

## Middle Hooks

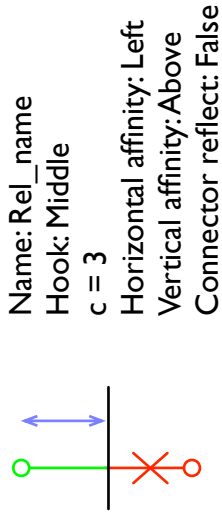
A UML association name is typically placed near the center like this:



So we can define a placement at some distance from the connector over the Middle Hook.



The distance away from the connector,  $C$ , is a positive value. We can also define horizontal and vertical affinity.



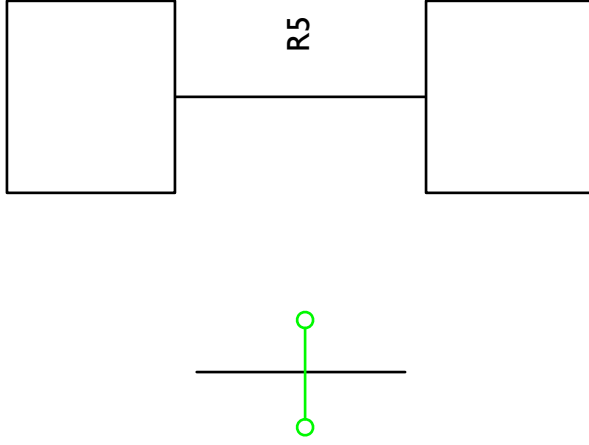
The reflect parameter indicates whether the space on the other side of the connector at distance  $c$  is available as an alternate label location. In this example, the reflected position is disabled, so the label may only appear above or right of the connector midpoint.

It could be argued that a lateral placement with respect to the midpoint might be desirable. But I doubt it's worth the added complexity. The only serious reason to displace a Label away from its Middle Hook would be to avoid an obstacle. But the stretch factor described in the collisions tech node addresses this issue. A Label may be displaced up to its specified stretch distance with respect to its reference Hook.

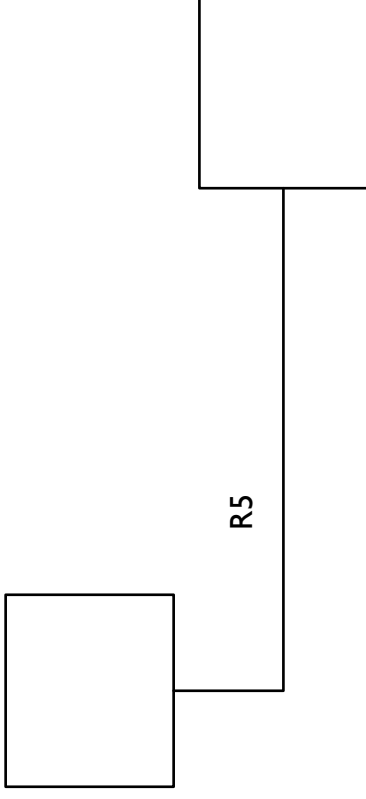
## Bend Hooks

Placements may be specified for any corners that appear in a Flexible Branch. (The actual positions will be available only if the Branch actually bends).

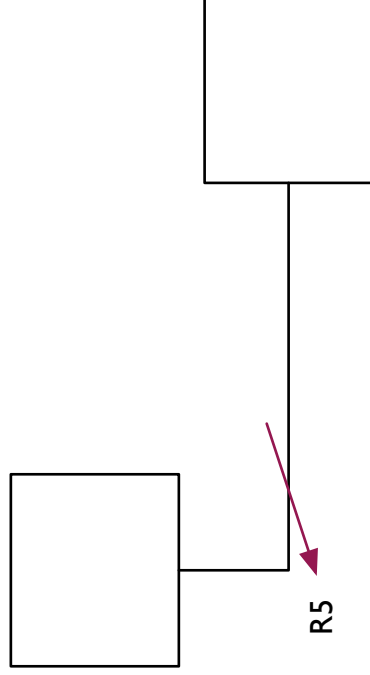
If a Binary Connector bends, a Label might be moved adjacent to one of the corners. Consider again the relationship name Label in a UML class model binary association:



Now if the Connector bends, the name Label will stay close to the middle of the Connector as shown.



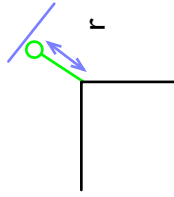
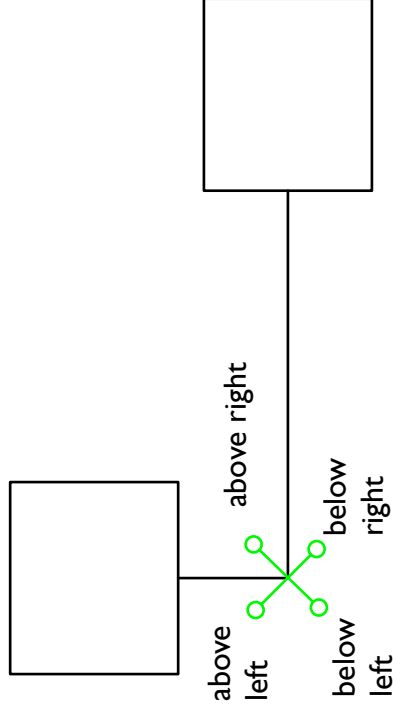
This is fine, but we may want to offer the possibility of pushing the name Label close to the bend point.



We now need a way to specify a position relative to any bend point on a Binary Connector.

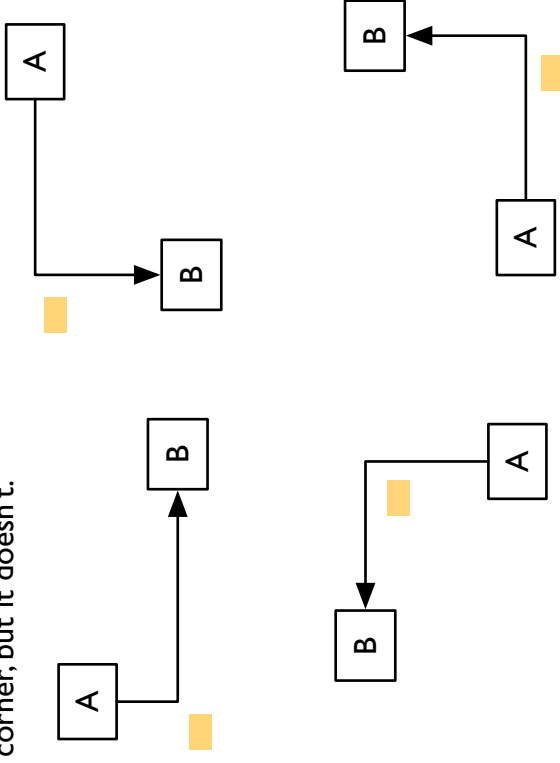
## Bend Hooks - first approach (fails!)

A simple way to specify position relative to a Bend Hook is to draw a radius ( $r$ ) at 45 degrees to the connector segments meeting at the corner. Combining this with horizontal and vertical affinity we get locations above-left, above-right, below-left and below-right.



To specify a position relative to one of the corners we just need the corner name and the radius distance ( $r$ ).

The nice thing about this scheme is that it keeps the h/v affinity scheme used with Terminator and Middle Hooks. Unfortunately, it doesn't play well with connector rotations. Consider a Label positioned with below-left affinity. We would expect it to remain in the outside corner, but it doesn't.

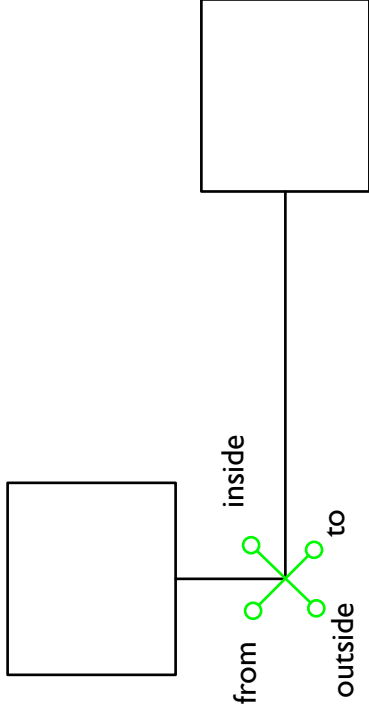


Of course we could just decide to accept this behavior, but the user probably thinks more in terms of how the label mates with the corner geometry (inside corner, outside corner, etc) rather than via h/v relationships to the two meeting connector segments which is less intuitive.

So let's try this again...

## Bend Hooks - 2nd approach, better

We will keep the (r) radius, but lose the affinity parameters and just use four distinct corner names: to, from, inside and outside. ("to" means the direction nearest the "to" Terminator).

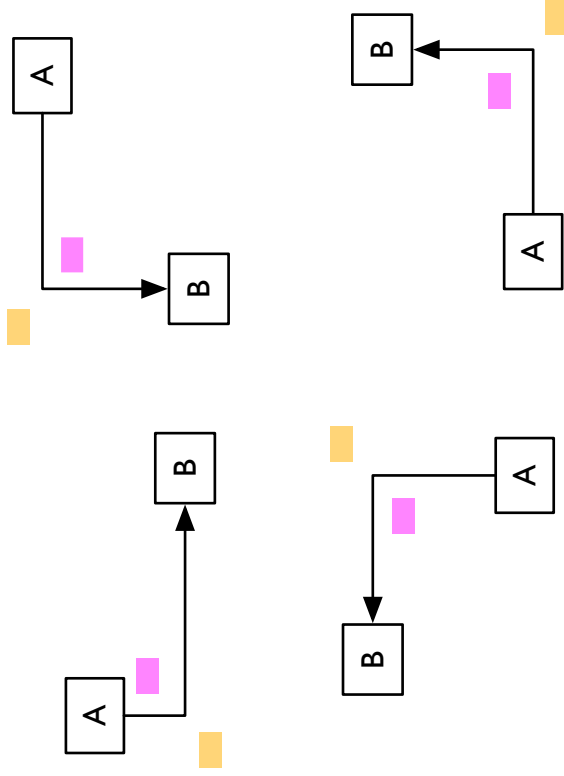


Now let's itemize the parameter value necessary to specify the placement of the relationship name Label at the outside corner (with the inside corner as an alternate).

Name: Rel\_name  
 Hook: Bend  
 c = 5  
 Corners: outside, inside

Note that we need a way to establish that one of the possible locations is the default (outside) while the rest are alternates (inside). This is an issue with Placements for the other Hook Types.

This reference scheme ensures logical positioning as a binary connector rotates.

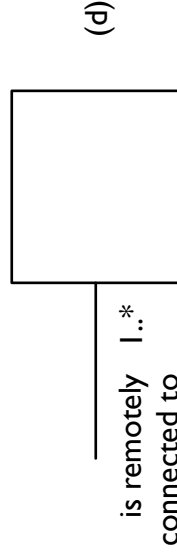
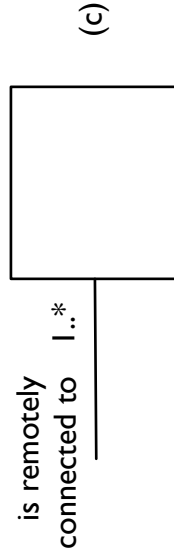
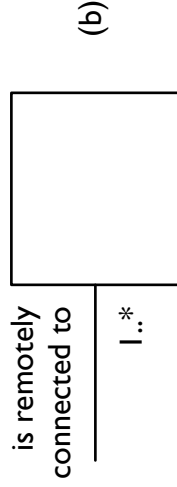
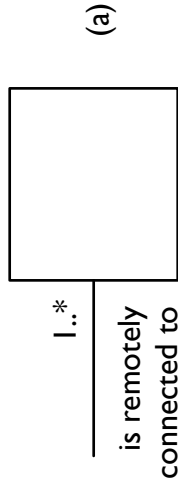


The bend references defined for a Label Specification apply to each Bend (if any exist) in a Connector. So if all four corner references are defined for a label specification and a corresponding connector bends twice, there will be eight possible locations for the label, four at each bend point.

If the Connector is straight (no bends), then none of the bend reference positions will be available. So, for every label specification with a bend corner reference, there should be at least one middle or terminator reference to ensure label placement under all conditions.

## Label collisions

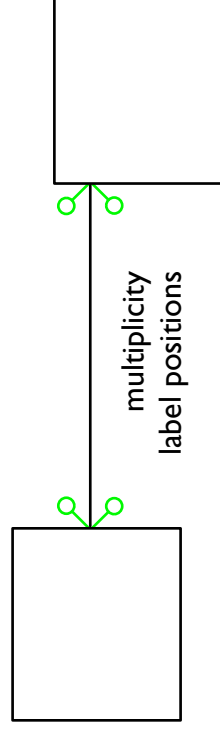
What do we do when more than one label may appear in the same general area? This happens on UML class model binary associations:



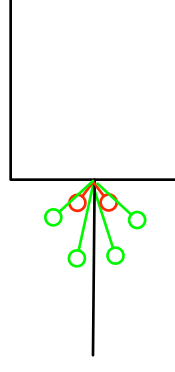
The multiplicity Label can appear on either side of the Connector near the Terminator. The verb phrase is typically placed in either location not used by the multiplicity label, on the other side of the Connector as shown in (a) and (b).

Sometimes, though, there is no room due to other obstacles, so the verb phrase is put on the same side as the multiplicity label, just next to it somewhere as shown in (c) and (d). Observe, however, that if the verb phrase is placed opposite the multiplicity label, it must take the position normally occupied by the multiplicity label as shown in example. So if the multiplicity label in (d) moves to the other side of the Connector, the verb phrase Label shifts right to take the multiplicity label's former place, reverting to case (a).

We can easily specify the locations of the multiplicity label with distances from the Terminator as shown previously:



The first idea is to just specify the verb phrase labels the same way with two positions coinciding with the multiplicity label positions like this:

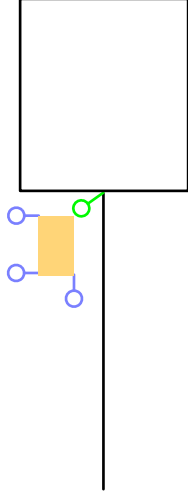


It's not going to work, though. The size of labels varies with font style and size. So labels close to one another may end up overlapping if we rely on positions relative only to a fixed point like the Terminator Hook. Also this doesn't account for shifting the verb phrase when the multiplicity label flips to the other side of the Connector.

## Label stacking

We need to specify the position of the verb phrase labels relative to the multiplicity labels. That way, when a multiplicity label moves over to the other side of a Connector the verb phrase will collapse into position. Also, if the font size changes, the verb phrase labels will move relative to new edge of the neighboring multiplicity label.

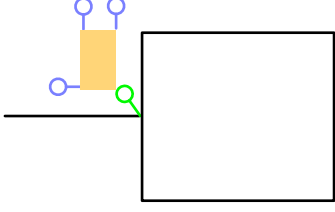
What we really want to do is to offer some sort of label stacking scheme for the rare case where labels can occupy each other's positions.



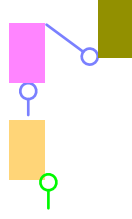
Here we can see that three positions for the verb phrase label are defined relative to three corners of the multiplicity label specification. These three are special label stacking placements, each relative to a corner of a base label, multiplicity, in this case.

A label placed relative to a stacking placement (blue) will drop back to the base placement (green) if the base label is somewhere else. If the base label is put back, the stacked label will be pushed out to one of the stack placements.

When the Binary Connector rotates, the stack references retain their integrity, even though the label itself is aligned to a new corner (lower left).



Even though there are no known examples where it is needed, it stands to reason that multiple levels of Label Specifications could be stacked relative to one another.

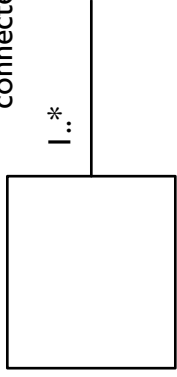


The mechanism sounds good, but we need to determine the exact parameters necessary to specify relative stacking positions. This is not as easy as it seems!

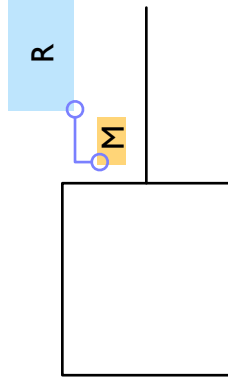
## Stack specification

Continuing with the UML class model binary association example, we want to specify an alternate position for the role label on top of the multiplicity label and slightly to the right.

I..\*  
is remotely  
connected to

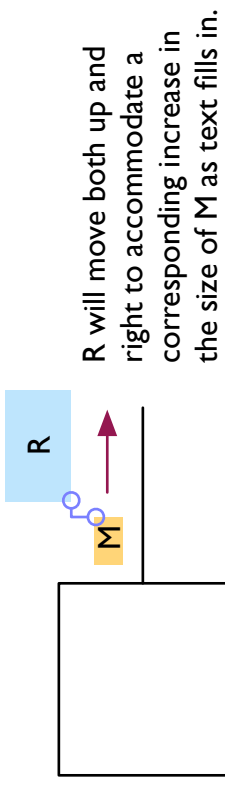


But the actual size of the text boxes is not known until the label text is entered. So we can pre-specify only the start corners of each Label.



The lower left corner of R may rise if M gets taller, but it won't budge in the horizontal direction.

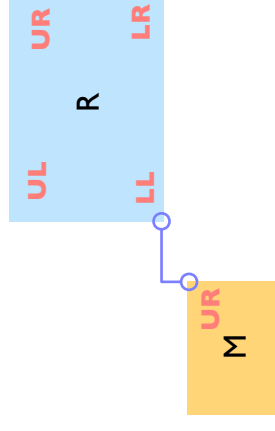
Had we chosen a different corner on M, lets say upper right, then R will move further right as the text fills out in M.



R will move both up and right to accommodate a corresponding increase in the size of M as text fills in.

We can leave it up to the diagram designer to choose the best corners based on desired behavior. But how will the corners and distances be specified?

Our first approach is the obvious approach of just naming the corners upper right, upper left, etc. Then we can specify distance relative to those corners.

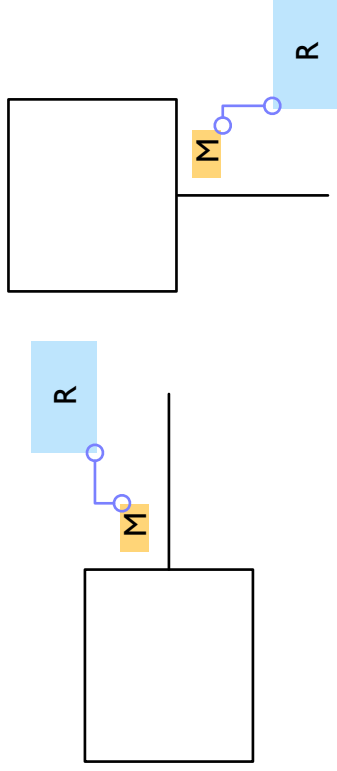


So we say that the stack relationship is something like  $M(UR) \rightarrow R(LL)$  with offsets  $x = 3$  and  $y = 1$ .

We know that the text will be aligned toward the Node when the connector segment is horizontal. Let's say that we offset the lower left corner of label Placement R from the upper left corner of Placement M. This means that no matter how wide or tall M or R become when filled with text, they will not overlap and remain at the same fixed distance.

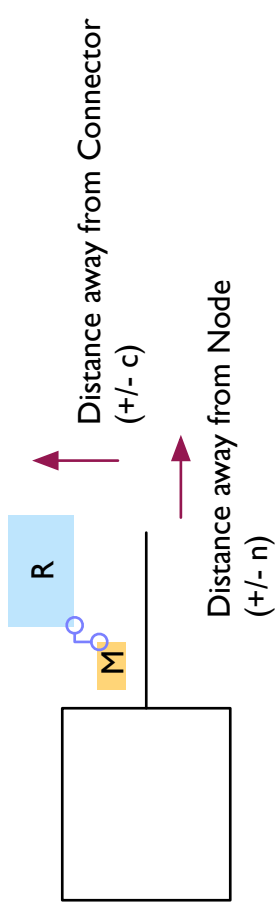
## Stack specification

Of course, this scheme works only as long as the node is on the left and the connector is horizontal. The references go bad as soon as we rotate the connector.

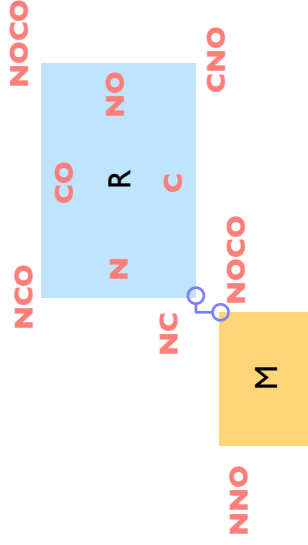


Now we have  $M(LR) \rightarrow R(UL)$  with offsets  $x = 1$  and  $y = -3$ . So we need another way to specify corners and distances in a rotation independent way.

As a Connector rotates in 90 degree increments, it is the relationship between the node face and the Connector that predominates. So we should specify distances relative to the Connector and Node directions and corners also relative to the Node and connector. Naming is a bit tricky, but let's give it a try.



Sides are named N (closest to node), C (closest to connector) and then NO (node outside) and CO (connector outside) for the opposite faces.



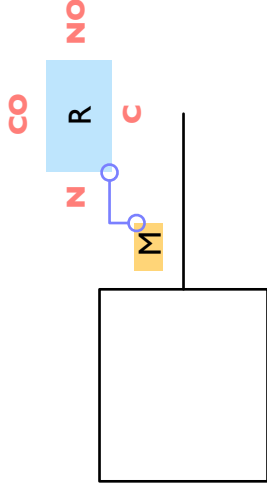
Now we rephrase our stack placement as  $M(NOCO) \rightarrow R(NC)$  with offsets  $n = 1$  and  $c = 3$ .

Let's test this system by rotating the connector and seeing if we can draw the labels in the correct locations.

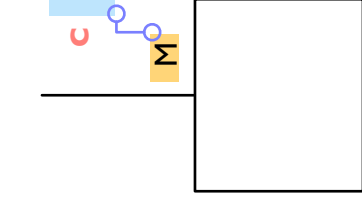
## Stack specification

All four rotations are tested below. We can see that the specification is now rotation invariant.

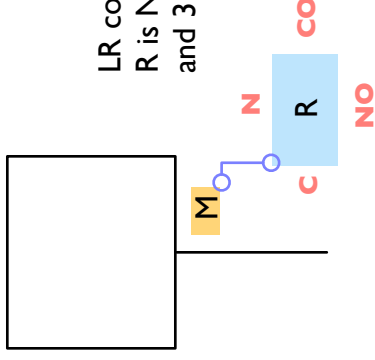
$M(NOCO) \rightarrow R(NC)$  with offsets  $n = 1$  and  $c = 3$ .



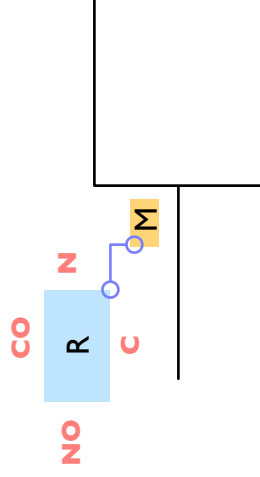
UL corner of M is NOCO and lower left of R is NC. R is 1 unit away from the connector and 3 units away from the node.



UR corner of M is NOCO and LL of R is NC. R is 1 unit away from the connector and 3 units away from the node.



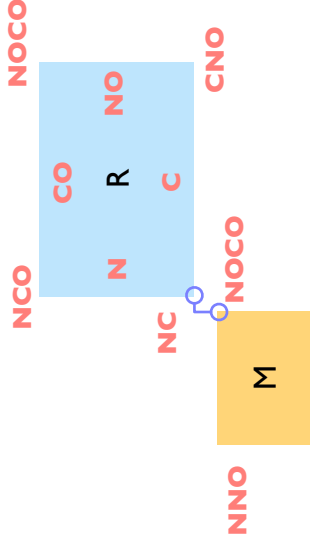
LR corner of M is NOCO and upper right of R is NC. R is 1 unit away from the connector and 3 units away from the node.



UL corner of M is NOCO and LR of R is NC. R is 1 unit away from the connector and 3 units away from the node.

Note that for the base label position in the stack, M, connector affinity is set to above / right.

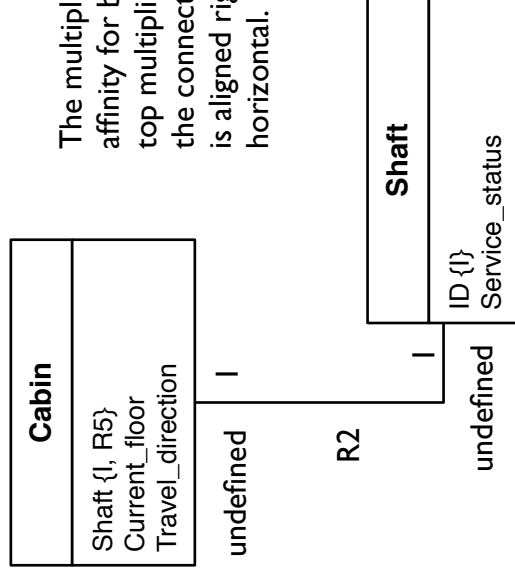
Also note that the while either value of  $n$  or  $c$  may be negative, they cannot both be negative!



## An example - default placement

Let's put all this together with an example. Since the UML class model binary association is the most complex labeling example we will use that.

We start out with two class nodes that we have just connected with a binary association. Default data specified for this connector type is filled into default label positions.



The multiplicity labels are both set to "1". The default affinity for both of these labels is "above" and "right". The top multiplicity is aligned left, toward the connector since the connector leg is vertical while the bottom multiplicity is aligned right, toward the node since that leg is horizontal.

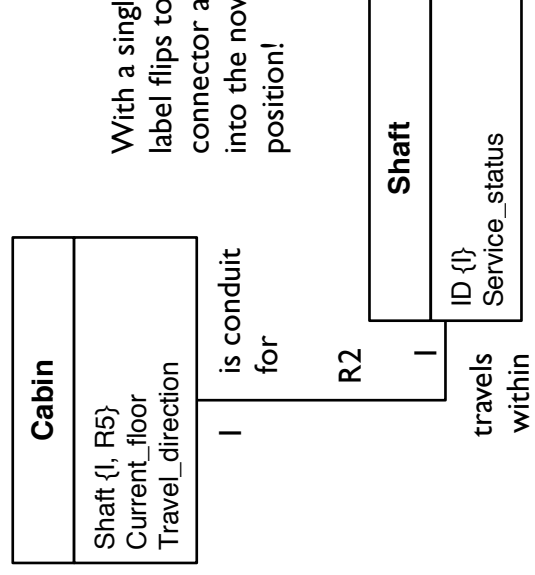
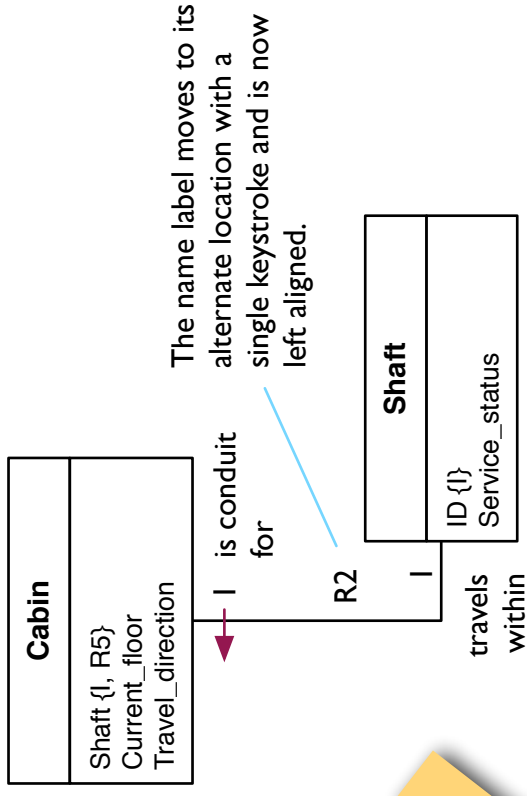
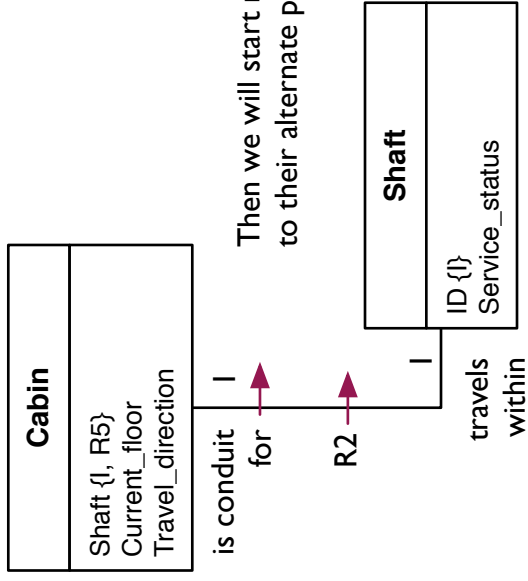
Finally, the verb phrase labels are filled in with the default text "undefined". Both labels have the default affinity of "bottom" and "left". Alignment rules are applied as for the multiplicity labels.

Now let's edit!

The name is autonumbered "R2", presumably this is the second connector drawn on the diagram. The default affinity is "below" and "left". And the label is default positioned relative to the connector midpoint. Since the leg is at this point, the label is right aligned - toward the connector.

## An example - edit

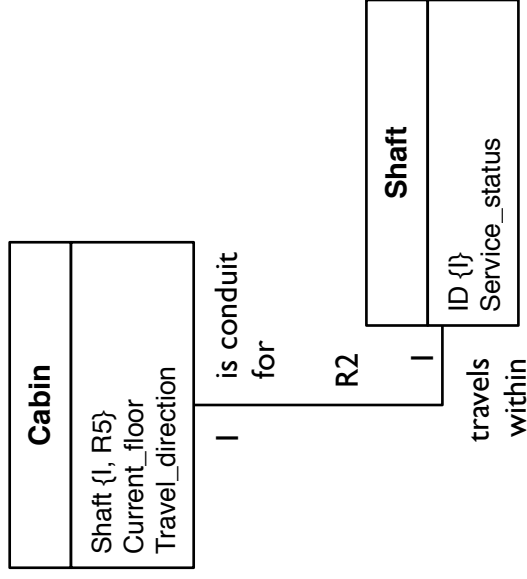
First we fill in the verb phrases. On each side the text fills in nicely right aligned with text moving down as enter keys are pressed to insert lines. There is no danger of node or connector overlapping while this happens.



With a single keystroke, the multiplicity label flips to the other side of the connector and its verb phrase collapses into the now vacant base stack position!

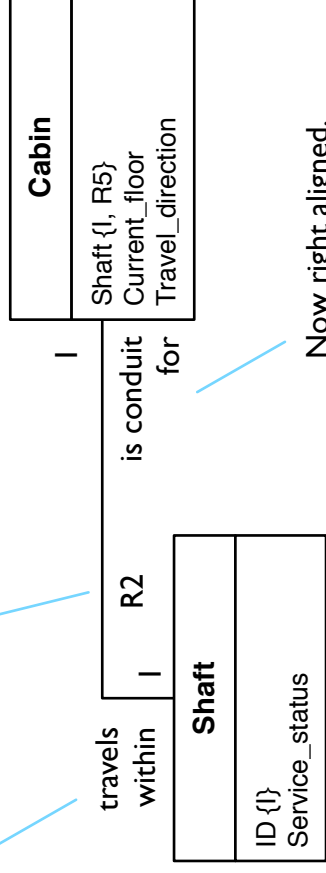
## An example - rotation

Now we apply a connector rotation and watch what happens to all of the labels.



Still right aligned toward vertical leg connector rather than node.

Name label is center aligned since leg is horizontal and reference is midpoint and top aligned toward connector.



Now right aligned.

The rotation and label re-alignments all happened on a single keystroke! (No magic - just simple rules and diagram/connector type specifications)

Try doing that with any existing UML tool!