Binder vs Filler terminology

Joop Ringelberg 18-01-22 Version: 1

Two sets of terms

Unfortunately, in the implementation of the PDR we use a different set of terms for connecting roles than in the language definition.

Representation

In the implementation we have a member in the representation of a role instance that is named `binding`. This points to the role that fills it.

Conversely, from roles we keep tabs on what other roles are filled by it. These are the `filledRoles`.[1]

Functions: binding, binder <Role>

We have a function for traversing the link in both directions:

- `binding` traverses from filled to filler[2];
- `binder <filled>` traverses from filler to filled[3].

Language: fills, filledBy

A role type definition is given partly in terms of the keyword `filledBy`. This uses a different metaphor. The converse would be `fills`.

Relation between the two; cardinality

The translation is simple:

Implementation	Language	Cardinality
Binder filled	fills filled	Multiple
Binding filler	filledBy filler	Just one

Obviously, a role can be filled by just one other role, but it can fill many. The following figure depicts the relations.

[1] Actually, the term is in Dutch: `gevuldeRollen`.
[2] We might want to rename `binding` to `filledBy`.
[3] We might want to rename `binder <RoleName>` to `fills <RoleName>`.

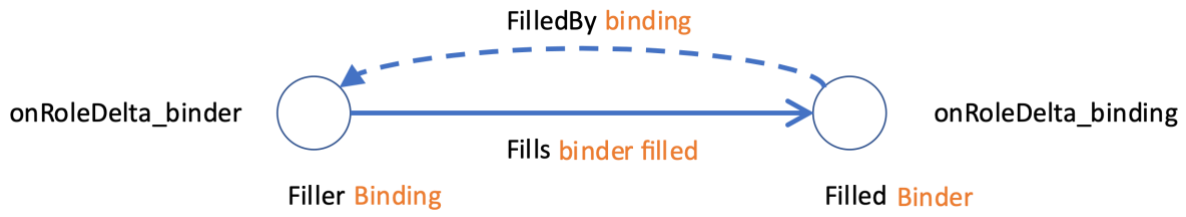

Figure 1. The two links that connect two roles.

Inverted queries

We store, with role types, inverted queries that lead back to the origin of perspectives (i.e. from the object of the perspective to its user role). We collect the inverted queries that start with a `binder` step in `onRoleDelta_binder`, and inverted queries that start with a `binding` step in `onRoleDelta_binding`.

The illustration shows how, for the same link between two roles, the inverted queries that traverse it are thus stored on opposite ends.

When we severe a connection

When the links between two roles are severed, we must trace inverted queries back to their origins to find (contexts with) users that should be informed. Two links will disappear (fills and filledBy), hence we must evaluate two sets of inverted queries:

- those in `onRoleDelta_binder` of the Filler;
- those in `onRoleDelta_binding` of the Filled.

Refer to Figure 1 for better understanding.

Direction: in and out of the context

A role R is attached to a context. R might fill roles in other contexts. We call a connection whose `fills` link departs from R *outgoing*.

Conversely, R might be filled by a role from another context. Therefore we call the connection whose `fills` link ends in R *incoming*.

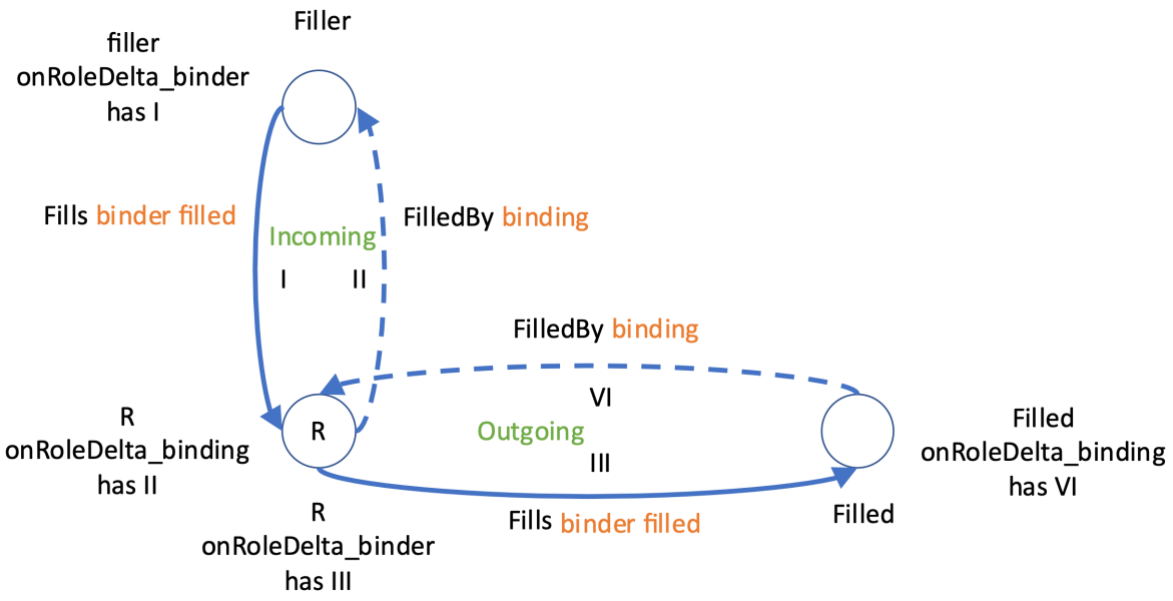

Figure 2 A single role in two relations: both as filled role and as filler role.

But, both inverted queries in:

- `onRoleDelta_binding` of R and
- `onRoleDelta_binder` of R

constitute links that move out of its context.

And, in terms of operations on R:

- `filledBy` moves out of its context;
- `fills` moves out of its context[4].

When we remove a role

When we completely remove a role, we have more work to do than when we just remove a binding.

Removing a role gives us a direction, because it is natural to reason from the context that the role is removed from. Again, we must trace inverted queries back to their origins to find (contexts with) users that should be informed.

However, we should now trace *all links in and out of the context* through the role to be removed. Referring to Figure 2, they are:

- those in `onRoleDelta_binder` of R;
- those in `onRoleDelta_binding` of Filled (and there may be many such roles);
- those in `onRoleDelta_binding` of the R;
- those in `onRoleDelta_binder` of Filler (just the one).

[4] In both cases: unless both roles come from the same context, of course.