Matching Activity Conditions to States

Matching conditions of an activity to a state depends on how containment and connectivity are defined in both.

Matching is done as follows:

1. Conditions in an activity are converted into Bigraph objects.
   1. The definition of a condition follows the one described by the incident meta-model.
   2. XML representation of a condition is converted to JSON object, which is then converted to Bigraph object.
2. The Bigraph object of a condition is compared against the signature of the Bigraph.
   1. If a condition has ***more connections*** for an entity than that in the signature (i.e. the number of defined *outernames* in a condition is more than that in the signature), then ***matching fails for all states***.
      1. For example, if in a condition an entity (or child) defines *three outernames* (i.e. connections) and in the bigraph signature the corresponding control to this entity defines only two outernames, then matching fails for all states. In the Bigraph object representation of the condition, the three outernames are defined, however, only two outernames are associated with controls, the rest (of outernames) are left empty i.e. outerName3:o <- {}
   2. If a condition has less connections for an entity, then:
      1. If **knowledge** about the connections is ***complete***, then ***matching fails for all states***.
      2. If **knowledge** is ***partial***, then the rest are defined as links (i.e. XX:**e**) in the bigraph object of the condition. In this case, matching moves to step 3.
3. Each condition is then compared against each state (already converted to Bigraph objects) using the bigraph matching, which is implemented by the method *match* that is provided by the *LibBig* library.
4. If the *match* method returns any object in the iterator (it returns an iterator, which shows where in a state the condition is match. Currently we are interested if there are anything return, we are not interested how many and where in a state), then ***there is a match*** and the state number is added to the condition (i.e. *predicate* object). Otherwise (i.e. if the iterator is empty) then ***there is no match*** between the condition and the current state.

Using the [match] function in the *LibBig* library:

The function depends in matching mainly on the containment relationships between the different entities. So it is important to specify *sites* where knowledge is not exact. For example, if a *Room* is an entity but we have no exact knowledge on what is inside the *Room* except a *SmartLight*, then it is necessary to add a *site* when creating a condition using the incident pattern meta-model.

For connectivity, matching is partial. For example:

* if an entity has 4 outernames (as arity in the signature) and in a condition there are 2 outernames defined and knowledge is partial and the rest are not *closed*, then defining only the 2 outernames by the Bigraph object will make the other two outernames *links* (i.e. XX:**e**), hence, it ***won’t match*** with the states that specify all 4 outernames.
  + Outernames by signature: 4
  + Outernames defined in condition: 2
  + Knowledge: partial, then:
    - we can define the other two outer name when creating the Bigraph object. In this situation, the condition ***will match*** with all states that define the 4 outernames as such (non are closed). It won’t matter that in a state outernames of an entity is associated with others. **What if in a state some outernames are closed (not found in the bigraph object)?** Probably this is not a possible case since closing an outername after adding it to a control will make the control representation a link i.e. XX:**e** in the link graph.
    - If the remaining two outernames are not defined then they are defined automatically by the Bigraph class as links (i.e. XX:**e**). in this case, they ***will not match*** with states that define outernames as XX:**o** (i.e. as outernames).
    - If we define the remaining outernames as ***closed*** (**before** adding them to controls or entities), then they will not match to states that define their outernames since they will not show in the Bigraph object. If we ***close*** outernames (**after** adding them to controls or entities), then they will be represented in the Bigraph object as links i.e. XX:**e.** **Will they match with states that close two of their outernames?** If they are represented as links, then yes(logically, needs testing).
  + For innernames, they can be added to a Bigraph object representation of a condition or a state directly i.e. they cannot be added to a node definition of a Bigraph object like outernames (as handles). Moreover:
    - An innername can be associated with an outername, which is linked to a control (e.g., XX:o <- {Hallway, inner1:i})
    - More than one innername can share the same outername.
    - An innername can have **only** **one** outername.

**Interesting note!** If all outernames are defined for two nodes separately (no common outernames), and in a state they do have common outernames, then this state will be matched according to the defined outernames for these two nodes. Should be checked is how matching is done based on this connectivity.

In other words, sharing the same outernames is treated the same as if all nodes had different outernames. For example:

Node1 has outernames o1, o2

Node2 has outernames o1

If a match is based on both nodes sharing o1 (i.e. o1 <- {node1, node2}), then it will match to the case above. However, also nodes that have different outernames will match (e.g., o1 <- {node1} o2 <- {node1} o3 <- {node2}) as long as they have the same number of outernames for each node (i.e. 2 for node1 and 1 for node2). This should be taken into consideration when matching!!

**Note:** Related to this, an interesting case where there was a shared outername between two nodes in the condition, and there were nodes with the same control in a state that share outernames. However, they did not match! Actually they matched when the outernames name changed to different names.

**Another note:** outernames will not match with edges (or links) i.e. XX:**o** doesn’t match XX:**e**

**The thing about connectivity in BRS:** after doing some tests around connectivity, it seems that the names, order, and shared connections has no effect on what reaction rules are invoked, on the condition that connections are represented as outernames (could be that they are defined as unbounded names, so they are only bound at execution time)., though Number of connections has an effect. For example if enter room reaction rule is the following:

Hallway{**a**,b}.Actor ! Room{**a**} -> Hallway{**a**,b} ! Room{**a**}.Actor

Then the naming and sharing of connections (defined as outernames in the above reaction rule has no effect on the evolution of the system). The above rule has the same effect as the following rule:

Hallway{**a**,b}.Actor ! Room{**z**} -> Hallway{**a**,b} ! Room{**z**}.Actor

So, what roles outernames play in BRS? how to represent connectivity in BRS?