Backward Search Implementation Details

Restrictions

- For a starting layer (the layer from which backward search starts), an element of the power set of the layer's services will be processed only if the services in the set produce at least 1 output parameter requested in the composition request.

  Reasons

  a. **Filtering out superfluous services:** A service set in the starting layer of a backward search represents the last layer of services in a service plan. If these services do not produce even 1 of the requested output parameters, they are not serving any requirements and can be removed. This is not applicable to the other layers as their service output parameters (even if they are not part of the composition request), might serve as inputs to the services in the later layers.

  b. **Eliminating the effort spent in processing invalid branches:** Since for every set of services in the starting layer, an exponentially growing branching and traversal process is triggered, eliminating unnecessary service sets at the beginning itself saves a considerable amount of effort during backward search.
  Additionally, the effort involved in pruning such branches and discarding duplicate plans that result from such pruning during the later stages is also reduced, thereby improving efficiency of the service composition process.

- A plan set generated by backward search is considered to be valid only if it contains more than 1 service.

  Reasons

  a. **Eliminating invalid service compositions:** A plan set comprising of a single service does not qualify as a service composition; it is merely an individual service, which can be retrieved by performing a search on the service repository.

Differences between Algorithm and Implementation

- In the backward search algorithm, a check is performed to ensure that all the inputs of a set of services in layer 0 are provided in the composition request. However, in the implementation, no such check is performed.

  Reason: The forward expansion process places only those services in layer 0 for which all the inputs are provided in the composition request. Therefore, another check for the same is not required in the backward search process. For the services in other layers, this check is not applicable because they receive their inputs partially or completely from services in their preceding layers.

- In the backward search algorithm, the service set is always obtained by taking an intersection between the set of services in the current layer and a predecessor service set provided as an input parameter. However, in the implementation, no such intersection is taken for the starting layer. All the services in that layer are together considered as the service set.

  Reason: Since the starting layer is the first layer to be processed during a search and its services are not predecessors to any service, the predecessor service set for the starting layer is a null set. Therefore, this method is not applicable to this layer. Also, taking an intersection with the null set would yield another null set, which would never be able to trigger a backward search.

- The loop for invoking the backward search algorithm for each layer of the search graph is executed by the service composition algorithm. However, in the implementation, this loop has also been incorporated in the backward search method itself.

  Reason: This has been done for better modularity and lower coupling. Since the complete backward search process is now contained within its own class, any future modifications in the process (if required) would not affect the service composition algorithm.