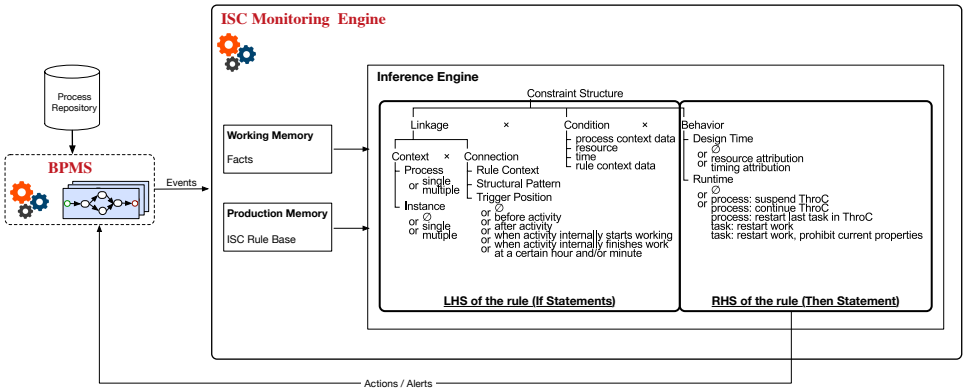

Figure 1: ISC Monitor: iUPC implementation

iUPC implementation inside the ISC Monitor As illustrated in Fig. 1, we highlight the iUPC concepts integrated within our implemented ISC monitor. Within the LHS, it is made clear that every attribute except behaviour is involved on the *if statement* side of the constraint rule. Linkage as well and condition are all implemented in terms of Rete nodes (alpha as well as beta nodes). What differs is the impact changing these attributes have on the global rete network.

Evaluation of an example change scenario: Delete Condition with Migration change strategy We start with the original ISC stating "When starting the read-out at 00:00, 99% of all meters should be read out within 6 hours and aggregated value should not exceed X." In this change scenario we will perform a Delete Condition change operation on the second condition, specifically on the requirement that the aggregated value should not exceed some limit X. The updated ISC then reads: "When starting the read-out at 00:00, 99% of all meters should be read out within 6 hours". Furthermore, we will apply the migration change strategy for this change scenario (cf. Fig. 2).

Static and Dynamic Impacts on the Alpha Network Regarding the static and dynamic impacts on the alpha network we observe that there are no impacts related to the inclusion of a router, thus there are no added routing paths based on instance_start_time related to the time of change (t_C). This is unique to the migration change strategy, as we strive to update all process instances to the ISC_new. For doing so we need to consider all *remaining* shared variables. The emphasis lies in the *remaining* shared variables due to performing a delete change operation. Thus we can directly remove those shared variables associated to the condition to be removed. In this case the shared variable is *accumulated_values*, and the associated event attribute: *readout*, which becomes unused due to the removal of the former. Since the connection stays intact, the facts decomposed from this event will still be added to the knowledge base. The static impact to the alpha network is thus the removal of alpha nodes related

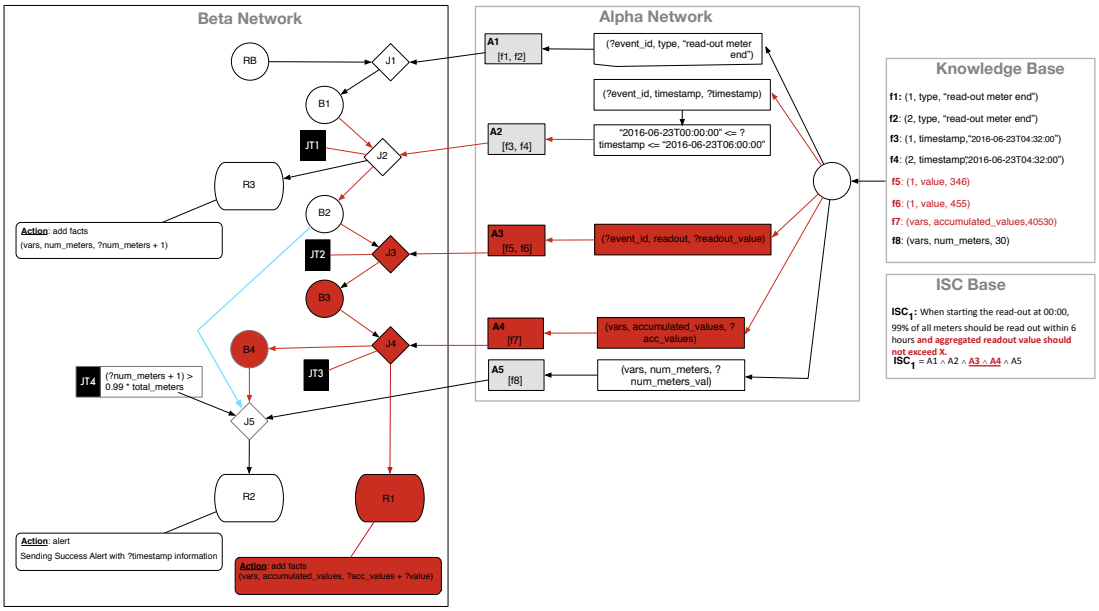

Figure 2: Change Impacts: Delete Condition with Migration change strategy

to the deleted condition (i.e., shared variables) as well as event data which all become unused. Regarding dynamic impact on the alpha network level, we have the removal of all working memory elements (WMEs) that were stored in $A4$ (i.e., $f7$) and $A3$ (i.e., $f5$ and $f6$). These are all the facts that are either related to *accumulated_values* or the event data attribute: *readout*.

Static and Dynamic Impacts on the Beta Network

Satically for the beta network, since we are removing $A4$ and $A3$, we can remove the paths leading to the corresponding action which verifies the limit X for *accumulated_values* has not been exceeded. For that purpose all the nodes and edges starting from both $A3$ and $A4$ are deleted: $J3$, $B3$, $J4$, $B4$, and finally $R1$. Additionally, dangling edges through this deletion process can be removed as well: the edge from $B2 \rightarrow J3$, as well as the edge from $B4 \rightarrow J5$. To make the structural change correct, not only deletions are performed on the beta network. A new edge from $B2 \rightarrow J5$ is added to complete the path for checking that 99% of all meter read outs have successfully finished within the required six hours. Regarding dynamic impacts on the resulting beta network, we can observe the deletion of all tokens (i.e., WMEs with successful joins) inside $B3$, $B4$ and $R1$. Due to this static change, future evaluations of this ISC will traverse the path starting from $J1$, and possibly ending with $R2$ via $B2 \rightarrow J5$. There are no explicit shared variable migrations due to no *remaining* shared variables to migrate. Compensation actions can be envisaged for the cases where past alerts regarding exceeded limits have been fired.

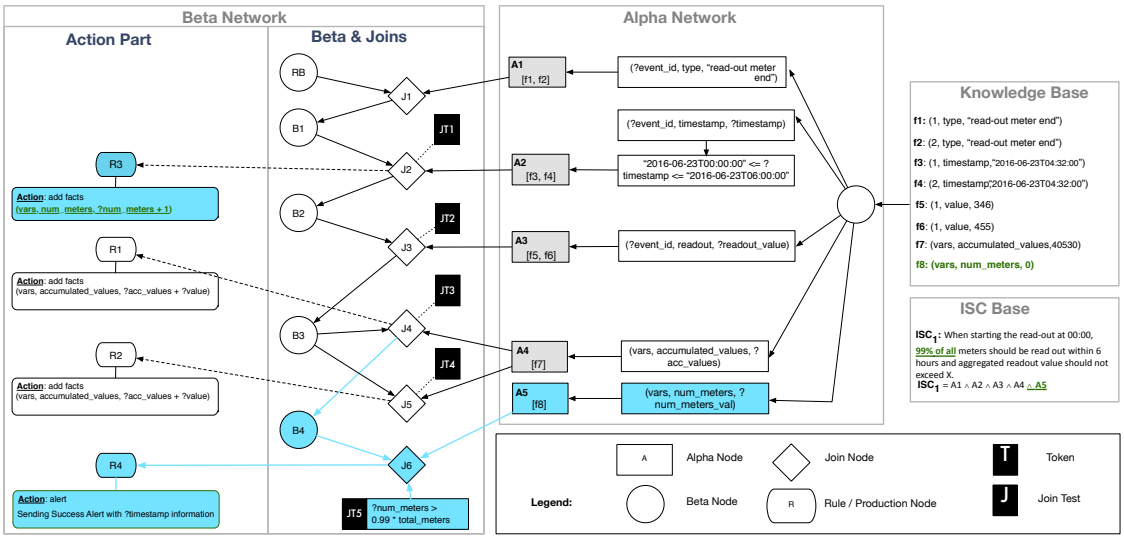

Figure 3: Change Impacts: Add Condition with Migration change strategy

Evaluation of an example change scenario: Add Condition with Migration change strategy

We start with the original ISC stating "When starting the read-out at 00:00, all meters should be read out within 6 hours and aggregated value should not exceed X." In this change scenario we will perform a Add Condition change operation before the first condition. The updated ISC then reads: "When starting the read-out at 00:00, *99% of* all meters should be read out within 6 hours and aggregated value should not exceed X". Furthermore, we will apply the migration change strategy for this change scenario (cf. Fig. 3). We can already imagine the effects as this scenario is a combination of both previous change scenarios: add condition will affect alpha and beta network in terms of the introduction of a new shared variable: *num_meters*. There is *no* versioning, and thus no routing logic to be implemented on the alpha as well as beta network. With the migration change strategy we only need to consider the existing shared variable: *accumulated_values*. In this case we will keep it intact as we can continue from the current state *accumulated_values* is in. The additional novel element is the requirement to add node $R3$, which is responsible for incrementing the newly added shared variable: *num_meters*.

Evaluation of an example change scenario: Update Condition with Migration change strategy

Starting from the original ISC stating "When starting the read-out at 00:00, 99% of all meters should be read out within 6 hours and aggregated value should not exceed X." For this change scenario we choose to update the ISC by changing the value to be successfully read out from 99% to 70%. This operation constitutes an update condition. Both statically and dynamically,

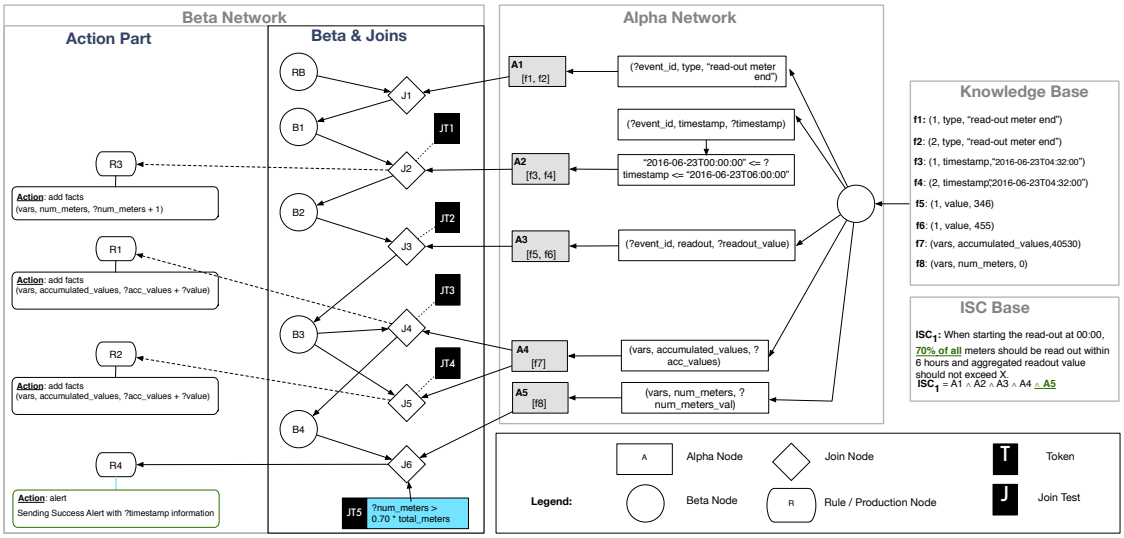

Figure 4: Change Impacts: Update Condition with Migration change strategy

there is no impact on the alpha network due to no alpha nodes being neither added nor deleted. In fact, we are reusing the node representing the condition being updated: $A5$. In contrast, a change strategy requiring a router such as the versioning or clean state strategies would indeed require a static change on this network for incorporating the router. The static impact on the beta network in turn, also reuses the existing join node for handling the existing condition $J6$, but triggers a change of the join test, which changes from 0.99 to 0.70 of total meters required to succeed. The dynamic impact occurs in consequence of this change, which retests all tokens that now successfully match this new condition. Previously successful matches (i.e. leading to firing of the associated action in $R4$) may have to be revoked through compensational actions. As can be seen in this scenario, update operations strive to reuse existing nodes without removing and re-adding them first.