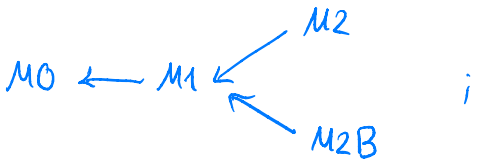
Modelling of the Robot Factory

Víctor Carrillo Redondo
Joseba Celaya Rodríguez
Luis Mata Aguilar

The modelling of the system was achieved via two contexts, C0 and Sensors, where the constants that define the system (n_arms, max_item_weight, min/max_box_capacity, on, off) and their relationship are introduced; and via four machines M0, M1, M2 and M2B that encapsulate the interaction between the different devices in the factory. It's important to note that the refining relationship between those four machines is the following:



;

and the behaviour modelled by each one may be summarized as follows:

- M0 is an initial approach to model the whole system in a very abstracted way: at this moment we do not really care about how boxes are replaced when the present one is close to be full and we don't worry about the synchronization of arms to place items in the box.
- The first refinement, M1, tries to talke into account that boxes cannot be automatically and immediately replaced and that we must model that physically replacing boxes may talke some time (but also that idle arms, i.e., arms that haven't picked any item from the conveyor belt yet, might

still pick new items while the box is being replaced; allowing the system to still be concurrent and distributed).

- The last refinement, carried out in both M2 and M2B, tries to tackle the synchronization of arms to place items in the box; again trying to ensure the concurrency of the system. It's also important to note that we might see M2 as an intermediate refinement between M1 and M2B, because in M2B we are doing a better separation between the environment and the controller (but we weren't able to completely refine M2 with M2B).

More in detail, and focusing on the requirements given, we have that most of them (EQP1, FUN 2-13) have already been addressed in M0, and that the two later refinements are focused on meeting requirements FUN 15 and FUN 14, respectively.

As mentioned above, in M0 we focused on modelling a physically abstracted version of the system and assumed, not only that boxes are immediately replaced and that arms do not need to synchronized, but also we assumed (and kept that assumption until the end) that the time that it takes for a sensor in the box to weight the item placed on it is negligible and that no event could interleave in that time span. Another simplification we assumed is that we don't really need to model the conveyor belt bringing new items.

Since one requirement, FUN 3, was that the conveyor belt can provide items continuously, we assume that when some arm that hasn't picked an item yet wants to pick one, it simply can. We thought of modelling it via an item pool or through a list where an arm could pick any of the first k (for k a constant of the model) items from it (like displaying the k-closest items to the arms), but we decided that this improvement was less important than the rest.

Requirements were addressed in M0 as follows:

EQP 1: through the constant n_arms and modelling the weight of the item picked by each arm as a function

arms : 1..n_arms → 0..max_item_weight, where the value 0 is used to represent that the given arm hasn't picked any item.

FUN 2: there is no event to return an item to the conveyor belt.

FUN 3: as explained above.

FUN 4: the controller can only know the weight of an item in event PICK_ITEM after an arm has been choosed to pick that item.

FUN 5,6,7: through the choice of constants and axioms in C0.

FUN 8: assuming that we can always pick items from the conveyor belt and through the action in PICK_ITEM that chooses a random weight for the "picked item".

FUN 9: as before, by modelling arms : 1..n_arms → 0..max_item_weight.

FUN 10: the guards in event PICK_ITEM tell us that if an arm doesn't already have an item, it can pick any item without needing to synchronize them.

FUN 11: the event that places an item in the box, PLACE_ITEM, can only happen if the item to place does fit on it.

FUN 12: by the guards of BOX_DEP.

FUN 13: after we removed a nearly full box, event BOX_ARR can always be fired. There's no limit on the number of boxes.

FUN 16: the controller updates the weight of the items in the box after placing every new item on it.