Pattern Extraction

# What Is required?

* An input is an incident model that is based on the incident pattern meta-model.
  + A model contains specific entities and relationships between them.

# Process to extract a pattern. It can be as follows

* First, do an abstraction round for the entities only. Define a set of entities SE (Specific Entity set), then do a function *Q(se) = ae*, where *se* belongs to SE and *ae* belongs to AE (Abstract Entity set). However, the abstract entity set (i.e. AE) is created from the Q function and can be refined/changed or different sets can be created that correspond to the Q function.
  + What is Q function? How should we define it?
  + We could introduce **abstraction levels** in the system meta-model. Thus, if an entity is abstracted to a level, then all entities in a condition (pre or post) will be abstracted to the same level. What is an abstraction level? An abstraction level can be defined by the inheritance and association relations.
  + How about properties of an entity? Indication of the abstraction level at which it can exist could be a solution.
* Second, do an abstraction of the conditions of concrete activities defined. How abstraction should be done? Define **rules**. Abstraction rules can be defined over the relationships (containment and connectivity) of BRS statements in conditions.

## Entity abstraction

The goal of this step is to find a *suitable* abstraction of the assets defined in an incident instance model.

For each asset in the incident instance model, abstract the asset to an entity that corresponds to a predefined level of abstraction in the system meta-model.

**System meta-model defines levels of abstraction**: initially define three levels of abstraction in the meta-model. abstraction level:

* Level 1: most abstract. For example, physical asset. Defines properties that can be at this level (e.g., name, connectivity, and containment)
* Level 2: less abstract (more concrete). For example, smart device. Defines what properties can be at this level (e.g., status).
* Level 3: least abstract (most concrete). For example, smart light. Again, defines what properties can be at this level (e.g., model number).

**Process for abstracting assets.**

*For* each asset:

1. Determine at what level the asset is (level 1, 2, or 3).
2. *If* asset is least abstract (most concrete), then abstract the asset to the previous level. For example, if an asset is at level 3, it is abstracted to level 2.
   * This includes creating new asset that is from the chosen abstraction level and define the properties that are defined in the original asset and that are part of the current chosen level
     + includes removing properties that belong to the previous level, unless the modeller specifies that a property should be kept during abstraction (could be used for learning). For example, remove the “model number” property if going from level 3 to level 2.
     + and giving new name to the new asset.
   * *The meta-model could specify preferred abstraction level (class)*
3. *Else If* an asset is not in the least abstract level (most concrete), then keep the asset as is, but change the name (retain mapping between new and old names).

**Output**: an incident instance model with abstracted entities.

Since the preconditions and postconditions are based on the connectivity and containment relationships between assets and that these relationships are defined at the highest abstraction levels, then we assume that replacing the original assets with the abstracted assets **will keep the meaning of the relationships**

## Conditions/Actions Abstraction

We need to define the set of rules on which actions (this means also the pre-/post-conditions) can be abstracted.

How to abstract containment relationship?

How to abstract connectivity relationship?