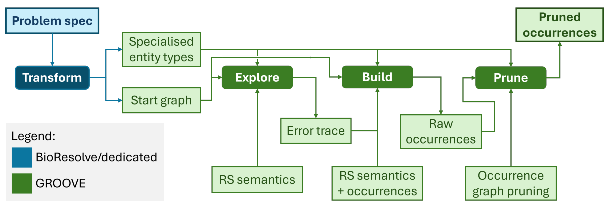


Figure 5: Reaction System exploration and analysis using GROOVE

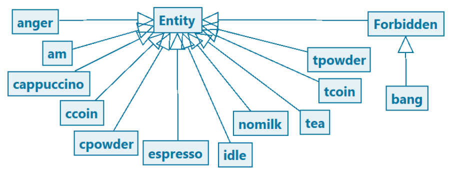
in question leads to a state in which a **Forbidden** entity is present (such as the bang entity in our toy example), we can also *prune* the occurrence graph, again using graph transformation, to keep only those rule occurrences and entity instances that directly contributed to the existence of the forbidden entity.

This gives rise to the tool chain depicted in Figure 5, the phases of which will be explained in some more detail in the remainder of this section.

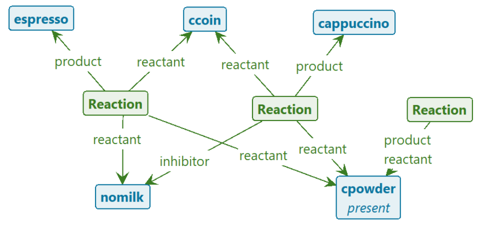
Transform. The first step is a text-to-model transformation from a problem specification in BioResolve syntax into GROOVE syntax. This is achieved by running the `main_do(rs2gts)` directive of BioResolve, which produces two artifacts: firstly, an additional type graph, complementary to the one shown in Figure 3, which specifies subtypes of **Entity** for all entities in the problem at hand (essentially for performance reasons: relying on dedicated types speeds up the matching step of GROOVE); and secondly (more importantly) a start graph in which the entire BioResolve system is encoded as suggested by Figure 3. For the example system, the additional types as well as two self-explanatory fragments of the start graph are shown in Figure 6.

Explore. The dynamics of Reaction Systems is encoded as a combination of two rules, context and react, which are scheduled to fire in alternation. The rule context encodes the simultaneous firing of all context processes (nondeterministically selecting an enabled **Step** from every **State** with a **Token**), whereas react encodes the (deterministic) simultaneous firing of all enabled **Reaction**s, while simultaneously erasing all **Entity**s that were not just produced. The production or erasure of an **Entity** is encoded through the creation or deletion of a *present* flag on a (persistent) **Entity** node, *not* by the creation or deletion of the node itself. In addition, to keep track of which nondeterministic choices were actually taken, the context rule marks the **Step**s that were selected with a *fired* flag, which is subsequently erased by the react rule.

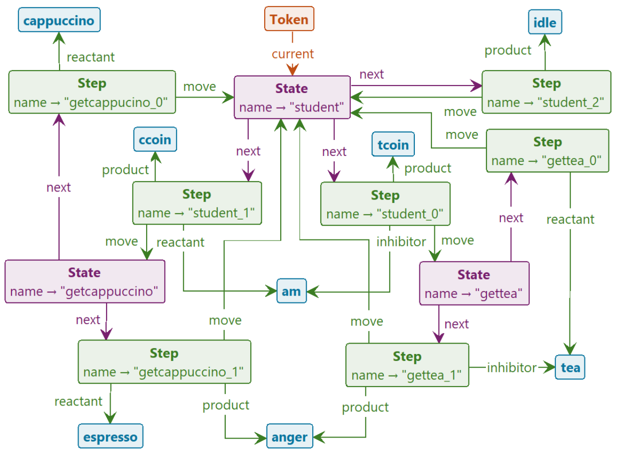
Figure 7 shows the first (and most intricate) of these rules, viz. the one for the context firing. This is



(a) Specialised entity types



(b) Start graph fragment: Three reactions from VM



(c) Start graph fragment: The Student context process

Figure 6: Graph representation of running example

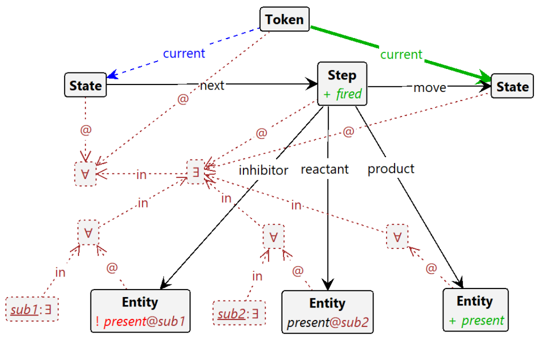


Figure 7: Rule for context firing