Department of Distributed and Dependable Systems
Technical report no. D3S-TR-2019-01
June 21, 2019

Formal Semantics of Component Ensembles

Tomas Bures, Petr Hnetynka

Abstract: In this report, we define a formal semantics of component ensembles.

1 Introduction

An *ensemble* is a group of components formed to perform joint goal or coordinate some activity. Members of an ensemble are established dynamically at runtime. An ensemble is determined by its *membership condition* – a predicate over components' types and knowledge. Ensembles can be hierarchically decomposed into further sub-ensembles. The semantics is that members of a sub-ensemble must be members of the parent ensemble too. This way, a top-level ensemble defines the goal of the system as a whole. A component can be a member of multiple ensembles at the same time, which naturally reflects the fact that a component may be part of a number of functionally orthogonal cooperations.

2 Ensemble Semantics

Definition 1. Component types and component instances We distinguish component types and component instances. Each component instance c is instantiated from a particular component type C. A component type C is associated with a set of attributes K that form the knowledge (i.e. the state) of a component instances of C. Each component instance c of type C is associated with a valuation of the knowledge – i.e. a function V_K that assigns each attribute $k \in K$ a particular value.

Definition 2. Ensemble types An ensemble type E is a tuple (P, R, G, M, U), where:

- P is a set of ensemble parameters.
- R is a set of roles of component roles in the ensemble. Each component role $r \in R$ is associated with function r_{dom} that for a given valuation of ensemble parameters V_P (see Definition 3) determines component instances that may be selected for the role (i.e. the powerset $2^{(r_{dom}(V_p))}$ is the domain for the role r).
- G is a set of sub-ensemble groups in the ensemble. Each sub-ensemble group $g \in G$ is associated with function gs_{dom} that for a given valuation of ensemble parameters V_P yields a set of tuples (E_i, V_P^i). Each of these tuples prescribes ensemble type and parameters for instantiation of a potential sub-ensemble in the sub-ensemble group g (more in Definition 3 below). We call the tuple (E_i, V_P^i) an ensemble instance specification.
- M is a membership condition that determines under what condition an ensemble is valid. The predicate M is parameterized by valuation of ensemble parameters, selection of component instances to each role r, and a set of sub-ensemble instances for each sub-ensemble group g.
- U is a utility function. Similarly, to M, it is parameterized by valuation of ensemble parameters, selection of component instances to each role r, and set of sub-ensemble instances for each sub-ensemble group g.

Definition 3. Ensemble instances An ensemble instance e of ensemble type $E = (P, R, G, M, U)$ is a tuple (V_P, V_R, V_G), where:

- V_P is a function that assigns a value to each parameter $p \in P$
- V_R is a function that to each component role $r \in R$ assigns a subset of $r_{dom}(V_P)$. This subset is the set of component instances selected as members of the ensemble instance (as part of the role r)
- V_G is a function that to each sub-ensemble group $g \in G$ assigns a set of ensemble instances I_g. Each ensemble instance e_j from I_g must comply with some ensemble instance specification (E_i, V_P^i) from the $gs_{dom}(V_P)$ associated with g.
  The projection from I_g to $gs_{dom}(V_P)$ does not have to be surjective (i.e. not all ensemble instance specifications in $gs_{dom}(V_P)$ have to be actually instantiated).
  Formally, for every $e_j = (V_P^j, V_R^j, V_G^j) \in I_g$ there exists an ensemble instance specification $(E_i, V_P^i) \in gs_{dom}(V_P)$ such that e_j complies with the specification (i.e. E_i is ensemble type of e_j and $V_P^j = V_P^i$).

The ensemble instance is valid only if all the following three conditions are true:

- the membership condition is satisfied – i.e. $M(V_P, V_R, V_G)$ is true
- all sub-ensemble instances are valid – i.e. $\forall g \in G, e_s \in V_G(g) : e_s$ is valid

- there is no cycle – i.e. an ensemble instance is not transitively its own sub-ensemble instance
- all component instances that are members of any sub-ensemble instance are also members of the ensemble instance

The utility of the ensemble instance is $U_e = U(V_P, V_R, V_G)$. Note that by being parameterized by V_G, the utility function U can aggregates utilities of sub-ensemble instances.

Definition 4. Root ensemble instance An ensemble instance is called root ensemble instance if it is not a sub-ensemble of another ensemble instance. We denote A the set of all root ensemble instances.

We assume function AS_{dom} which yields a set ensemble instance specifications (E_i, V_P^i). Each such specification determines how to instantiate one potential root ensemble instance – i.e. for each root ensemble instance $e_j = (V_P^j, V_R^j, V_G^j) \in A$ there exists an ensemble instance specification $(E_i, V_P^i) \in AS_{dom}$ such that e_j complies with the ensemble instance specification.

For the sake of instantiation of ensembles, we also associate each tuple $(E_i, V_P^i) \in AS_{dom}$ with a component instance $c \in C$. This component c, is responsible for instantiating the corresponding ensemble instance e. We call the component c an initiator for ensemble instance e.

Definition 5. Optimal instantiation of ensembles A valid ensemble instance $e = (V_P, V_R, V_G)$ of type E is *optimal* (with respect to ensemble instance specification (E, V_P) that specifies how to instantiate the ensemble instance) if there is no other valid ensemble instance $e^{'} = (V_P^{'}, V_R^{'}, V_G^{'})$ that complies with the ensemble instance specification (E, V_P) and $U_e < U_{e'}$. Ensembles in a system are optimally instantiated (with respect to AS_{dom}) if for each ensemble instance specification $(E_i, V_P^i) \in AS_{dom}$, one of the following conditions hold:

- There exists a corresponding $e_j = (V_P^j, V_R^j, V_G^j) \in A$ such that complies with (E_i, V_P^i) and is optimal.
- There exists no valid $e_j = (V_P^j, V_R^j, V_G^j)$ that would comply with (E_i, V_P^i)