RCL 2000 has two nondeterministic functions, oneelement and allother. The oneelement(X) function allows us to get one element x_i from set X. We usually write oneelement as OE. Multiple occurrences of OE(X) in a single *RCL 2000* statement all select the same element x_i from X. With allother(X) we can get a set by taking out one element. We usually write allother as AO. These two nondeterministic functions are related by context, because for any set S, $\{OE(S)\} \cup AO(S) = S$, and at the same time, neither is a deterministic function.

In order to illustrate how to use these two functions to specify role-based constraints, we take the requirement of the static separation of duty (SOD) property which is the simplest variation of SOD. For simplicity assume there is no role hierarchy (otherwise replace roles by roles*).

Requirement: *No user can be assigned to two conflicting roles.* In other words, conflicting roles cannot have common users. We can express this requirement as below.

Expression: $|roles(OE(U)) \cap OE(CR)| \leq 1$

OE(CR) means a conflicting role set and the function roles(OE(U)) returns all roles that are assigned to a single user OE(U). Therefore this statement ensures that a single user cannot have more than one conflicting role from the specific role set OE(CR). We can interpret the above expression as saying that if a user has been assigned to one conflicting role, that user cannot be assigned to any other conflicting role. We can also specify this property in many different ways using *RCL 2000*, such as $OE(OE(CR)) \in roles(OE(U)) \Rightarrow AO(OE(CR)) \cap roles(OE(U)) = \phi$ or $user(OE(OE(CR))) \cap user(AO(OE(CR))) = \phi$.

The expression $|roles(OE(sessions(OE(U)))) \cap OE(CR)| \leq 1$ specifies dynamic separation of duties (DSOD) applied to active roles in a single session as opposed to static separation applied to user-role assignment. Dynamic separation applied to all sessions of a user is expressed by $|roles(sessions(OE(U))) \cap OE(CR)| \leq 1$.

A permission-centric formulation of separation of duty is specified as $roles(OE(OE(CP))) \cap roles(AO(OE(CP))) = \phi$. The expression roles(OE(OE(CP))) means all roles that have a conflicting permission from, say cp_i, and roles(AO(OE(CP))) stands for all roles that have other conflicting permissions from the same conflicting permission set cp_i. This formulation leaves open the particular roles to which conflicting permissions are assigned but requires that they be distinct. This is just a sampling of the expressive power of *RCL 2000* discussed in Section 4.

RCL 2000 system functions do not include a time or state variable in their structure. So we assume that each function considers the current time or state. For example, the sessions function maps a user u_i to a set of current sessions that are established by user u_i. Elimination of time or state from the language simplifies its formal semantics. *RCL 2000* thereby cannot express history or time-based constraints. It will need to be extended to incorporate time or state for this purpose.