STS-IQ: Properties of the design

Pro1-4 are used to verify goals related properties.

pro1_achievement_violation_root(A, G) :- aims(A, G), not achieved(A, G).

pro1: states that the model should not include any goal that is not achieved from the perspective of the actor, who aims for it. This property is used to quickly verify the IQ requirements model, i.e., if it holds for all goals, the model is correct and consistent.

pro2_achievement_violation_leaf(A, G) :- is_responsible(A, G), prevented(G).

Pro2: states that the model should not include any goal that is not achieved from the perspective of the actor, who is responsible for its achievement, which enables for detecting goals that have been prevented from being achieved.

pro3_dele_goal_no_trust_violation(A, B, G) :- deleChain(A, B, G), not trustedChain(A, B, G).

Pro3: states that the model should not include any goal delegation/delegation chain, unless there is a trust chain between the delegator and the delegatee, or there is a compensation for the lack of trust/distrust among them, since delegation without trust leave the delegator with no guarantee that its goal will be achieved.

pro4_delegate_goal_violation(A, B, G) :- deleChain(A, B, G), not can_achieve(B, G), is_responsible(C, G), not deleChain(B, C, G).

Pro4: states that goals should be only delegated to actors, who have the capability to achieve them either by themselves or they have a valid delegation to an actor who has such capability.

Pro5-6 are used to verify information availability properties.

pro5_availability_violation(A, I) :- need(A, I), info(I, _), not has(A, I).

Pro5: states that actors should have all information that is required for the achievement of the goals they are responsible for, i.e., this property is specialized for detecting information unavailability related issues.

pro6_provision_violation(A,I) :- prvChain(_, _, _, A, I), need(B, I), not need(A, I), not prvTChain(A, B, I).

Pro6: states that information should be only provided to actors, who require them either for achieving their goals, or they have valid provision to an actor who requires such information.

pro7-10 are used to verify information permissions related properties.

pro7_perm_unavailable(P, A, I) :- need_perm(P, A, I), not has_perm(P, A, I).

pro7: states that actors should have all permissions they need to perform their activities.

pro8_perm_has_violation(P, A, I) :- has_perm(P, A, I), not need_perm(P, A, I), not own(A, I).

pro8: states that permissions should be only delegated to actors, who require them for performing their activities. Only information owners may have permissions they do not require.

pro9_perm_dele_violation(p, A, I) :- dele_perm_chain(A, B, P, R, M, S, I), not has_perm(Perm, A, I).

pro9: states that the model should not include actors who delegate permissions that they do not have.

pro10_perm_violation_owner(Perm, A, I) :- own(B,I), has_perm(Perm, A, I), not trustedPerm(B, A, Perm ,I), not own(A,I).

pro10: states that the model should not include actors who have permissions, and there is no trust chain between such actors and information owner concerning the delegated permissions.

pro11-19 verify IQ related properties.

pro11_inaccurate_produce(A, I) :- producer(A, I, T), not accurate_produce(A,I).

pro11: states that the model should not include produced information that is not accurate from its producer's perspective.

pro12_inaccurate_send(A, I) :- sender(T, A, B, I), hasT(B, I, _), not accurate_send(T, A, B, I).

pro12: state that the model should not include sent information that is inaccurate at its destination from the perspective of its sender.

pro13_incomplete_send(A, I) :- sender(T, A, B, I), not complete_send(T, A, B, I).

pro13: state that the model should not include sent information that is incomplete at its destination from the perspective of its sender.

pro14_invalid_send(A, I) :- sender(T, A, B, I), not valid_send(T, A, B, I).

pro14: state that the model should not include sent information that is invalid at its destination from the perspective of its sender.

pro15_inaccessible_read(A, I) :- reader(_, _, _, A, I), not has_perm(r, A, I).

pro15: state that the model should not include information that is inaccessible from the perspective of its reader.

pro16_inaccurate_read(A, I) :- reader(Type, POU, BType, A, I), hasT(A, I, T), not accurate_read(A, I).

pro16: state that the model should not include information that is inaccurate from the perspective of its reader.

pro17_incomplete_read(A, I) :- reader(Type, PoU, BType, A, I), not complete_read(A, I).

pro17: state that the model should not include information that is incomplete from the perspective of its reader.

pro18_invalid_read(A, I) :- reader(Type, POU, BType, A, I), not valid_read(A, I).

pro18: state that the model should not include information that is invalid from the perspective of its reader.

pro19_inconsistent_read(A, I) :- reader(Type, POU, BType, A, I), not consistent_read(A, I).

pro19: state that the model should not include information that is inconsistent from the perspective of its reader.

pro20-22 verify monitoring related properties.

pro20_extra_monitoring(A, B, I) :- info(I,_), monitorChain(A, B, I), not threat_source(B, I).

pro20: states that the model should not include a monitoring relation between information reader and its producer unless the last can be considered as a threat source for such information.

pro21_extra_monitoring(A, B, G) :- goal(G), monitorChain(A, B, G), trustChain(trust, A, B, G).

pro21: states that the model should not include a monitoring relation between a goal delegator and a delegatee, if there exists a trust chain between them considering the delegated goal.

pro22_extra_monitoring_perm(A, B, Perm) :- trust_perm_chain(A, B, [trust], [trust], [trust], [trust], I), monitorPermChain(A, B, [mon], [mon], [mon], [mon], I).

pro22: states that the model should not include a monitoring relation between a permission delegator and a delegatee, if there is a trust relation among them considering the delegated permission.

pro23-24 verify separation of duty related properties.

pro23_conflict_roles_produce(A, I) :- play(A, R1), play(A, R2), conflict_roles(R1, R2, p, I), producer(A, I, _).

pro23: state that the model should not include any agent that plays conflicting roles in terms of producing information

pro24_conflict_roles_read(A, I) :- play(A, R1), play(A, R2), conflict_roles(R1, R2, r, I), reader(_, _, _, A, I).

pro24: state that the model should not include any agent that plays conflicting roles in terms of reading information.