MODULE *IrsaOperator*

EXTENDS *TLC*, *Naturals*, *Sequences*

Caveat :

- distributed consensus is not displayed here (operator *SDK* handles this for us)
- multi *CR* mechanism is not displayed here (simple scoping is enough to avoid collisions)
- we also assume the specs are valid

CONSTANTS

$NULL$, dummy constant

$\_workers$ the set of reconcile loops

VARIABLES

$irsa$, the iamroleserviceaccount *CR*

$policy$, the policy *CR*

$role$, the role *CR*

sa, the *serviceAccount* to be created

$awsPolicy$, the *IAM* policy on aws

$awsRole$, the *IAM* role on aws

wq, the *k8s* workqueue

$workers$, the workers (concurrent reconcile loops)

$modified$ the *k8s* resources modified during an action (simulates the watch mechanism)

$vars \triangleq \langle irsa, policy, role, sa, awsPolicy, awsRole, wq, workers, modified\rangle$

the different requests

$iReq \triangleq$ "irsa"

$pReq \triangleq$ "policy"

$rReq \triangleq$ "role"

$saReq \triangleq$ "sa"

$pendingSt \triangleq$ "pending"

$valid\_states \triangleq \{NULL, pendingSt\}$

$TypeOk \triangleq$

$\land\ irsa.st \in valid\_states$

$\land\ policy.st \in valid\_states$

$\land\ role.st \in valid\_states$

$\land\ sa.st \in valid\_states$

$\land\ \forall w \in \text{DOMAIN } workers : workers[w].req \in \{NULL, iReq, pReq, rReq, saReq\}$

$\land\ awsRole.arn \in \{NULL, \text{"roleARN"}\}$

$\land\ awsPolicy.arn \in \{NULL, \text{"policyARN"}\}$

$Init \triangleq$

$\land\ irsa = [st \mapsto NULL,$
$\quad saName \mapsto \text{"saName"},$
$\quad stmt \mapsto \text{"statement"},$
$\quad roleARN \mapsto NULL,$
$\quad policyARN \mapsto NULL$
$\quad]$

$\land\ policy = [st \mapsto NULL,$
$\quad stmt \mapsto NULL,$
$\quad awsPolicyArn \mapsto NULL$

```
            ]
  ∧ role = [st ↦ NULL,
             saName ↦ NULL,
             roleArn ↦ NULL,
             policyArn ↦ NULL,
             policiesAttached ↦ FALSE
               NB : this last flag is not yet in the implementation.
               It's needed to avoid missing the attached policies
             ]
  ∧ sa = [ st ↦ NULL,
           name ↦ NULL,
           roleArn ↦ NULL
         ]
  ∧ awsPolicy = [arn ↦ NULL]  union already created as expected & different
  ∧ awsRole = [arn ↦ NULL, attachedPolicy ↦ NULL]  union already created as expected & different
  ∧ modified = ⟨iReq⟩
  ∧ wq = [dirty ↦ {}, processing ↦ {}, queue ↦ ⟨⟩]  we start with an IrsaRequest in the dirty set
  ∧ workers = [w ∈ _workers ↦ [idle ↦ TRUE, req ↦ NULL]]
```

k8s workqueue

```
Enqueue(r) ≜  sequence of modified resources, simulating the watch mechanism
  ∧  modified' = modified ∘ r

  tla spec of the k8s workqueue algorithm
  see : https://github.com/kubernetes/client-go/blob/a57d0056dbf1d48baaf3cee876c123bea745591f/util/workqueue/queue.go ≠ L65
Add ≜
  ∧ modified ≠ ⟨⟩
  ∧ modified' = Tail(modified)
  ∧ LET e ≜ Head(modified) IN
      IF e ∈ wq.dirty
       THEN
          ∧ UNCHANGED ⟨irsa, policy, role, sa, awsPolicy, awsRole, workers, wq⟩
       ELSE
          ∧ IF e ∉ wq.processing
              THEN wq' = [wq EXCEPT !.dirty = wq.dirty ∪ {e}, !.queue = Append(wq.queue, e)]
              ELSE wq' = [wq EXCEPT !.dirty = wq.dirty ∪ {e}]
          ∧ UNCHANGED ⟨irsa, policy, role, sa, awsPolicy, awsRole, workers⟩

Get(w) ≜
  ∧ workers[w].idle
  ∧ workers[w].req = NULL
  ∧ wq.queue ≠ ⟨⟩
  ∧ LET head ≜ Head(wq.queue) IN
      ∧  workers' = [workers EXCEPT ![w] = [idle ↦ FALSE, req ↦ head]]
      ∧  wq' = [wq EXCEPT !.queue = Tail(wq.queue), !.dirty = wq.dirty \ {head}, !.processing = wq.processing ∪ {head}]
  ∧ UNCHANGED ⟨awsPolicy, awsRole, irsa, modified, policy, role, sa⟩

Done(w) ≜
  ∧ workers[w].idle
  ∧ workers[w].req ≠ NULL
```

```
    $\land$ $workers' = [workers$ EXCEPT $![w] = [idle \mapsto$ TRUE, $req \mapsto NULL]]$
    $\land$ LET $r \triangleq workers[w].req$IN
        IF $r \in wq.dirty$
          THEN $wq' = [wq$ EXCEPT $!.processing = wq.processing \setminus \{r\}, !.queue = Append(wq.queue, r)]$
          ELSE $wq' = [wq$ EXCEPT $!.processing = wq.processing \setminus \{r\}]$
    $\land$ UNCHANGED $\langle awsPolicy, awsRole, irsa, modified, policy, role, sa\rangle$
```

the expected states when a resource has converged

```
$IrsaComplete \triangleq$
    $\land policy.st \neq NULL$
    $\land role.st \neq NULL$
    $\land sa.st \neq NULL$

$policyComplete \triangleq$
    $\land policy.st \neq NULL$
    $\land policy.stmt \neq NULL$
    $\land policy.awsPolicyArn \neq NULL$

$roleComplete \triangleq$
    $\land role.st \neq NULL$
    $\land role.saName \neq NULL$
    $\land role.roleArn \neq NULL$
    $\land role.policyArn \neq NULL$
    $\land role.policiesAttached$

$saComplete \triangleq$
    $\land sa.st \neq NULL$
    $\land sa.name \neq NULL$
    $\land sa.roleArn \neq NULL$
```

operator specific actions

```
 $NB$ : update policy not displayed yet
$CreatePolicy(w) \triangleq$
    $irsa$ controller
    $\land workers[w].idle =$ FALSE
    $\land workers[w].req = iReq$
    $\land policy.st = NULL$ policy doesn't exist
    $\land policy' = [policy$ EXCEPT $!.st =$ "pending", $!.stmt = irsa.stmt]$
    $\land workers' = [workers$ EXCEPT $![w].idle =$ TRUE$]$
    $\land Enqueue(\langle pReq, iReq\rangle)$
    $\land$ UNCHANGED $\langle awsPolicy, awsRole, irsa, role, sa, wq\rangle$

$CreateRole(w) \triangleq$
    $irsa$ controller
    $\land workers[w].idle =$ FALSE
    $\land workers[w].req = iReq$
    $\land role.st = NULL$ role doesn't exist
    $\land role' = [role$ EXCEPT $!.st =$ "pending", $!.saName = irsa.saName]$
    $\land workers' = [workers$ EXCEPT $![w].idle =$ TRUE$]$
    $\land\ Enqueue(\langle rReq,\ iReq\rangle)$
    $\land$ UNCHANGED $\langle awsPolicy,\ awsRole,\ irsa,\ policy,\ sa,\ wq\rangle$

 if it has one, we'll try to update it, not shown yet
$PolicyHasNoARN(w)\ \triangleq$
    policy controller
    $\land\ workers[w].idle = \text{FALSE}$
    $\land\ workers[w].req = pReq$
    $\land\ policy.awsPolicyArn = NULL$
    $\land$ IF $awsPolicy.arn = NULL$
        THEN $\land\ awsPolicy' = [awsPolicy$ EXCEPT $!.arn =$ "policyARN"$]$
             $\land\ Enqueue(\langle pReq\rangle)$
             $\land$ UNCHANGED $\langle awsRole,\ irsa,\ policy,\ role,\ sa,\ workers,\ wq\rangle$
        ELSE $\land\ policy.awsPolicyArn = NULL$
             $\land\ policy' = [policy$ EXCEPT $!.awsPolicyArn = awsPolicy.arn]$
             $\land\ Enqueue(\langle pReq\rangle)$
             $\land$ UNCHANGED $\langle awsRole,\ awsPolicy,\ irsa,\ role,\ sa,\ workers,\ wq\rangle$

$RoleHasNoRoleARN(w)\ \triangleq$
    role controller
    $\land\ workers[w].idle = \text{FALSE}$
    $\land\ workers[w].req = rReq$
    $\land\ role.roleArn = NULL$
    $\land$ IF $awsRole.arn = NULL$
        THEN $\land\ awsRole' = [awsRole$ EXCEPT $!.arn =$ "roleARN"$]$
             $\land$ UNCHANGED $\langle awsPolicy,\ irsa,\ policy,\ role,\ sa,\ workers,\ wq\rangle$
        ELSE $\land\ role' = [role$ EXCEPT $!.roleArn = awsRole.arn]$
             $\land$ UNCHANGED $\langle awsPolicy,\ awsRole,\ awsPolicy,\ irsa,\ policy,\ sa,\ workers,\ wq\rangle$
    $\land\ Enqueue(\langle rReq\rangle)$

$RoleHasNoPolicyARN(w)\ \triangleq$
    role controller
    $\land\ workers[w].idle = \text{FALSE}$
    $\land\ workers[w].req = rReq$
    $\land\ role.policyArn\ = NULL$
    $\land\ policy.awsPolicyArn \neq NULL$
    $\land\ role' = [role$ EXCEPT $!.policyArn = policy.awsPolicyArn]$
    $\land\ Enqueue(\langle rReq\rangle)$
    $\land$ UNCHANGED $\langle awsPolicy,\ awsRole,\ awsPolicy,\ irsa,\ policy,\ sa,\ workers,\ wq\rangle$

$RoleHasPolicyARN(w)\ \triangleq$
    role controller
    $\land\ workers[w].idle = \text{FALSE}$
    $\land\ workers[w].req = rReq$
    $\land\ role.policyArn\ \neq NULL$
    $\land\ role.roleArn \neq NULL$
    $\land\ \neg role.policiesAttached$
    $\land\ awsRole.attachedPolicy = NULL$
    $\land\ awsRole' = [awsRole$ EXCEPT $!.attachedPolicy = role.policyArn]$
    $\land\ role' = [role$ EXCEPT $!.policiesAttached = \text{TRUE}]$
    $\land\ Enqueue(\langle rReq\rangle)$
    $\land$ UNCHANGED $\langle awsPolicy, irsa, policy, sa, workers, wq \rangle$

$CreateServiceAccount(w) \triangleq$
     irsa controller
    $\land$ $workers[w].idle =$ FALSE
    $\land$ $workers[w].req = iReq$
    $\land$ $sa.st = NULL$
    $\land$ $roleComplete$
    $\land$ $policyComplete$
    $\land$ $sa' = [sa$ EXCEPT $!.st =$ "pending", $!.name = irsa.saName, !.roleArn = role.roleArn]$
    $\land$ $Enqueue(\langle saReq, iReq \rangle)$
    $\land$ UNCHANGED $\langle awsPolicy, awsRole, irsa, policy, role, workers, wq \rangle$

 the following actions just "swallow" events when there's nothing to do on the resource
$IrsaAllDone(w) \triangleq$
    $\land$ $workers[w].idle =$ FALSE
    $\land$ $workers[w].req = iReq$
    $\land$ $IrsaComplete$
    $\land$ $workers' = [workers$ EXCEPT $![w].idle =$ TRUE$]$
    $\land$ UNCHANGED $\langle awsPolicy, awsRole, irsa, policy, role, sa, wq, modified \rangle$

$PolicyAllDone(w) \triangleq$
    $\land$ $workers[w].idle =$ FALSE
    $\land$ $workers[w].req = pReq$
    $\land$ $policyComplete$
    $\land$ $workers' = [workers$ EXCEPT $![w].idle =$ TRUE$]$
    $\land$ UNCHANGED $\langle awsPolicy, awsRole, irsa, policy, role, sa, wq, modified \rangle$

$RoleAllDone(w) \triangleq$
    $\land$ $workers[w].idle =$ FALSE
    $\land$ $workers[w].req = rReq$
    $\land$ $roleComplete$
    $\land$ $workers' = [workers$ EXCEPT $![w].idle =$ TRUE$]$
    $\land$ UNCHANGED $\langle awsPolicy, awsRole, irsa, policy, role, sa, wq, modified \rangle$

$SaAllDone(w) \triangleq$
    $\land$ $workers[w].idle =$ FALSE
    $\land$ $workers[w].req = saReq$
    $\land$ $saComplete$
    $\land$ $workers' = [workers$ EXCEPT $![w].idle =$ TRUE$]$
    $\land$ UNCHANGED $\langle awsPolicy, awsRole, irsa, policy, role, sa, wq, modified \rangle$

 the whole state converged
$Termination \triangleq$
    $\land$ $\forall w \in$ DOMAIN $workers : workers[w].idle =$ TRUE $\land\ workers[w].req = NULL$
    $\land$ $IrsaComplete$
    $\land$ $roleComplete$
    $\land$ $policyComplete$
    $\land$ $saComplete$
    $\land$ $awsPolicy.arn \neq NULL$
    $\land$ $\land$ $awsRole.arn \neq NULL$
      $\land$ $awsRole.attachedPolicy \neq NULL$
    $\land$ UNCHANGED $vars$
```

Spec

$$\begin{array}{l} \mathit{Actions} \triangleq \\ \quad \lor \mathit{Add} \\ \quad \lor \exists\, w \in \_\mathit{workers} : \begin{array}[t]{l} \lor \mathit{Get}(w) \\ \lor \mathit{Done}(w) \\ \lor \mathit{CreatePolicy}(w) \\ \lor \mathit{CreateRole}(w) \\ \lor \mathit{CreateServiceAccount}(w) \\ \lor \mathit{PolicyHasNoARN}(w) \\ \lor \mathit{RoleHasNoRoleARN}(w) \\ \lor \mathit{RoleHasNoPolicyARN}(w) \\ \lor \mathit{RoleHasPolicyARN}(w) \\ \lor \mathit{IrsaAllDone}(w) \\ \lor \mathit{PolicyAllDone}(w) \\ \lor \mathit{RoleAllDone}(w) \\ \lor \mathit{SaAllDone}(w) \end{array} \end{array}$$

$$\begin{array}{l} \mathit{Fairness} \triangleq \\ \quad \land \mathrm{WF}_{\mathit{vars}}(\mathit{Add}) \\ \quad \land \mathrm{WF}_{\mathit{vars}}(\mathit{Termination}) \\ \quad \land \forall\, w \in \_\mathit{workers} : \begin{array}[t]{l} \land \mathrm{WF}_{\mathit{vars}}(\mathit{Get}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{Done}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{CreatePolicy}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{CreateRole}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{CreateServiceAccount}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{PolicyHasNoARN}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{RoleHasNoRoleARN}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{RoleHasNoPolicyARN}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{RoleHasPolicyARN}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{IrsaAllDone}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{PolicyAllDone}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{RoleAllDone}(w)) \\ \land \mathrm{WF}_{\mathit{vars}}(\mathit{SaAllDone}(w)) \end{array} \end{array}$$

$$\begin{array}{l} \mathit{Next} \triangleq \\ \quad \lor \mathit{Actions} \\ \quad \lor \mathit{Termination} \end{array}$$

$$\begin{array}{l} \mathit{Spec} \triangleq \\ \quad \land \mathit{Init} \\ \quad \land \Box[\mathit{Next}]_{\mathit{vars}} \\ \quad \land \Box\, \mathit{TypeOk} \\ \quad \land \mathit{Fairness} \end{array}$$

Expectations

Safety

$$\begin{array}{l} \mathit{NoConcurrentProcessingOfSameResource} \triangleq \\ \quad \Box \forall\, w \in \textsc{domain}\ \mathit{workers} : \begin{array}[t]{l} \lor \mathit{workers}[w].\mathit{idle} \\ \lor \mathit{workers}[w].\mathit{req} \notin \{\mathit{workers}[x].\mathit{req} : x \in \textsc{domain}\ \mathit{workers} \setminus \{w\}\} \end{array} \end{array}$$

Liveness

$$
\begin{array}{l}
\mathit{TerminationIsTheLastAction} \triangleq \\
\quad \Box\textsc{enabled}\ \mathit{Termination} \leadsto \begin{array}{l}\land\ \textsc{enabled}\ \mathit{Termination}\\ \land\ \neg\textsc{enabled}\ \mathit{Actions}\end{array}
\end{array}
$$

$\textsc{theorem}\ \mathit{Spec} \Rightarrow \mathit{NoConcurrentProcessingOfSameResource}$
$\textsc{theorem}\ \mathit{Spec} \Rightarrow \mathit{TerminationIsTheLastAction}$