Contents

CONTEXT c0_service_registry

SETS

S set of all possible services

E set of all possible endpoints to assign to the different services

CONSTANTS

S1
S2
S3
S4
S5
E1
E2
E3
E4
E5
E6

AXIOMS

ax1: $partition(S, \{S1\}, \{S2\}, \{S3\}, \{S4\}, \{S5\})$

ax2: $partition(E, \{E1\}, \{E2\}, \{E3\}, \{E4\}, \{E5\}, \{E6\})$

END

CONTEXT c1_service_registry_w_health_check
EXTENDS c0_service_registry
SETS

HCA set of all possible health check apis to assign to each endpoint

CONSTANTS

HCA1
HCA2
HCA3
HCA4

AXIOMS

ax3: $partition(HCA, \{HCA1\}, \{HCA2\}, \{HCA3\}, \{HCA4\})$

END

MACHINE m0_service_registry
SEES c0_service_registry
VARIABLES

services represents the set of microservices in the registry

endpoints represents the endpoints of the microservices in the resgistry

last_endpoint_query_result represents the last response obtained from querying a service's endpoints

INVARIANTS

inv1: $services \subseteq S$

inv2: $endpoints \in services \rightarrow \mathbb{P}(E)$

inv4: $last\_endpoint\_query\_result \in \mathbb{P}(E)$

EVENTS

Initialisation

begin

init1: $services := \varnothing$

init2: $endpoints := \varnothing$

init3: $last\_endpoint\_query\_result := \varnothing$

end

Event register ⟨ordinary⟩ $\widehat{=}$

any

a_service

where

grd1: $a\_service \in S \setminus services$

can only register a new service

then

act1: $services := services \cup \{a\_service\}$

act2: $endpoints(a\_service) := \varnothing$

end

Event unregister ⟨ordinary⟩ $\widehat{=}$

any

a_service

where

grd1: $a\_service \in services$

can only unregister existing services

then

act1: $endpoints := \{a\_service\} \lhd endpoints$

domain substraction

act2: $services := services \setminus \{a\_service\}$

end

Event add_endpoint ⟨ordinary⟩ $\widehat{=}$

any

a_service

an_endpoint

where

grd1: $a\_service \in services$

can only add an endpoint to a previously registered service

grd2: $an\_endpoint \in E \setminus union(ran(endpoints))$

two services can not share endpoints

then

act1: $endpoints(a\_service) := endpoints(a\_service) \cup \{an\_endpoint\}$

end

Event remove_endpoint ⟨ordinary⟩ $\widehat{=}$

any

a_service

an_endpoint

where

grd1: $a\_service \in services$

can only remove an endpoint from a previously registered service

grd2: $an\_endpoint \in endpoints(a\_service)$
can only remove a registered endpoint of a service

then

act1: $endpoints(a\_service) := endpoints(a\_service) \setminus \{an\_endpoint\}$

end

Event query_endpoints ⟨ordinary⟩ $\widehat{=}$

any

a_service
result

where

grd1: $a\_service \in S$
grd2: $result \subseteq E$
grd3: $a\_service \in services \Rightarrow result = endpoints(a\_service)$
grd4: $a\_service \notin services \Rightarrow result = \varnothing$

then

act1: $last\_endpoint\_query\_result := result$

end

END

MACHINE m1_service_registry_w_health_check
REFINES m0_service_registry
SEES c1_service_registry_w_health_check
VARIABLES

services represents the set of microservices in the registry
endpoints represents the endpoints of the microservices in the resgistry
health_apis represents the api for health checking a service endpoint
last_endpoint_query_result represents the last response obtained from querying a service's endpoints
last_health_query_result represents the last response obtained from querying an endpoint's health apis

INVARIANTS

inv1: $services \subseteq S$
inv2: $endpoints \in services \rightarrow \mathbb{P}(E)$
inv4: $last\_endpoint\_query\_result \in \mathbb{P}(E)$
inv3: $health\_apis \in union(ran(endpoints)) \rightarrow HCA$
inv5: $last\_health\_query\_result \in \mathbb{P}(HCA)$

EVENTS
Initialisation
begin
init1: $services := \varnothing$
init2: $endpoints := \varnothing$
init3: $last\_endpoint\_query\_result := \varnothing$
init4: $health\_apis := \varnothing$
init5: $last\_health\_query\_result := \varnothing$
end
Event register ⟨ordinary⟩ $\widehat{=}$
extends register
any
$a\_service$
where
grd1: $a\_service \in S \setminus services$
can only register a new service
then
act1: $services := services \cup \{a\_service\}$
act2: $endpoints(a\_service) := \varnothing$
end
Event unregister ⟨ordinary⟩ $\widehat{=}$
extends unregister
any
$a\_service$
where
grd1: $a\_service \in services$
can only unregister existing services
then
act1: $endpoints := \{a\_service\} \lhd endpoints$
domain substraction
act2: $services := services \setminus \{a\_service\}$
act3: $health\_apis := endpoints(a\_service) \lhd health\_apis$
domain substraction
end
Event add_endpoint ⟨ordinary⟩ $\widehat{=}$
extends add_endpoint
any
$a\_service$
$an\_endpoint$
a_health_check_api
where

grd1: $a\_service \in services$
can only add an endpoint to a previously registered service
grd2: $an\_endpoint \in E \setminus union(ran(endpoints))$
two services can not share endpoints
grd3: $a\_health\_check\_api \in HCA$

then

act1: $endpoints(a\_service) := endpoints(a\_service) \cup \{an\_endpoint\}$
act2: $health\_apis(an\_endpoint) := a\_health\_check\_api$

end

Event remove_endpoint ⟨ordinary⟩ $\widehat{=}$
extends remove_endpoint

any

$a\_service$
$an\_endpoint$

where

grd1: $a\_service \in services$
can only remove an endpoint from a previously registered service
grd2: $an\_endpoint \in endpoints(a\_service)$
can only remove a registered endpoint of a service

then

act1: $endpoints(a\_service) := endpoints(a\_service) \setminus \{an\_endpoint\}$
act2: $health\_apis := \{an\_endpoint\} \lhd health\_apis$
domain substraction

end

Event query_endpoints ⟨ordinary⟩ $\widehat{=}$
extends query_endpoints

any

$a\_service$
$result$

where

grd1: $a\_service \in S$
grd2: $result \subseteq E$
grd3: $a\_service \in services \Rightarrow result = endpoints(a\_service)$
grd4: $a\_service \notin services \Rightarrow result = \varnothing$

then

act1: $last\_endpoint\_query\_result := result$

end

Event query_health_apis ⟨ordinary⟩ $\widehat{=}$

any

an_endpoint
result

where

grd1: $an\_endpoint \in E$
grd2: $result \subseteq HCA$
grd3: $an\_endpoint \in union(ran(endpoints)) \Rightarrow result = \{health\_apis(an\_endpoint)\}$
grd4: $an\_endpoint \notin union(ran(endpoints)) \Rightarrow result = \varnothing$

then

act1: $last\_health\_query\_result := result$

end

END

MACHINE m2_service_registry_w_health_check
REFINES m1_service_registry_w_health_check
SEES c1_service_registry_w_health_check
VARIABLES

services represents the set of microservices in the registry
endpoints represents the endpoints of the microservices in the resgistry
health_apis represents the api for health checking a service endpoint
last_endpoint_active_query_result represents the last response obtained from querying a service's endpoints
last_health_query_result represents the last response obtained from querying an endpoint's health apis
time represents current time, always moving forward
unavailable_endpoints represents all endpoints that are not available for consumption

INVARIANTS

inv1: $services \subseteq S$
inv2: $endpoints \in services \rightarrow \mathbb{P}(E)$
inv4: $last\_endpoint\_active\_query\_result \subseteq last\_endpoint\_query\_result$
inv3: $health\_apis \in union(ran(endpoints)) \rightarrow HCA$
inv5: $last\_health\_query\_result \in \mathbb{P}(HCA)$
inv6: $time \in \mathbb{N}$
inv7: $unavailable\_endpoints \subseteq union(ran(endpoints))$

EVENTS
Initialisation
begin
init1: $services := \varnothing$
init2: $endpoints := \varnothing$
init3: $last\_endpoint\_active\_query\_result := \varnothing$
init4: $health\_apis := \varnothing$
init5: $last\_health\_query\_result := \varnothing$
init6: $time := 0$
init7: $unavailable\_endpoints := \varnothing$
end
Event register ⟨ordinary⟩ $\widehat{=}$
extends register
any
$a\_service$
where
grd1: $a\_service \in S \setminus services$
can only register a new service
then
act1: $services := services \cup \{a\_service\}$
act2: $endpoints(a\_service) := \varnothing$
end
Event unregister ⟨ordinary⟩ $\widehat{=}$
extends unregister
any
$a\_service$
where
grd1: $a\_service \in services$
can only unregister existing services
then
act1: $endpoints := \{a\_service\} \lhd endpoints$
domain substraction
act2: $services := services \setminus \{a\_service\}$
act3: $health\_apis := endpoints(a\_service) \lhd health\_apis$
domain substraction
act4: $unavailable\_endpoints := unavailable\_endpoints \setminus endpoints(a\_service)$
end

Event add_endpoint ⟨ordinary⟩ $\widehat{=}$
extends add_endpoint
any
$a\_service$
$an\_endpoint$
$a\_health\_check\_api$
where
grd1: $a\_service \in services$
can only add an endpoint to a previously registered service
grd2: $an\_endpoint \in E \setminus union(ran(endpoints))$
two services can not share endpoints
grd3: $a\_health\_check\_api \in HCA$
then
act1: $endpoints(a\_service) := endpoints(a\_service) \cup \{an\_endpoint\}$
act2: $health\_apis(an\_endpoint) := a\_health\_check\_api$
end
Event remove_endpoint ⟨ordinary⟩ $\widehat{=}$
extends remove_endpoint
any
$a\_service$
$an\_endpoint$
where
grd1: $a\_service \in services$
can only remove an endpoint from a previously registered service
grd2: $an\_endpoint \in endpoints(a\_service)$
can only remove a registered endpoint of a service
then
act1: $endpoints(a\_service) := endpoints(a\_service) \setminus \{an\_endpoint\}$
act2: $health\_apis := \{an\_endpoint\} \lhd health\_apis$
domain substraction
act3: $unavailable\_endpoints := unavailable\_endpoints \setminus \{an\_endpoint\}$
end
Event query_endpoints ⟨ordinary⟩ $\widehat{=}$
refines query_endpoints
any
a_service
result2
where
grd1: $a\_service \in S$
grd2: $result2 \subseteq E$
grd3: $a\_service \in services \Rightarrow result2 = endpoints(a\_service) \setminus unavailable\_endpoints$
grd4: $a\_service \notin services \Rightarrow result2 = \varnothing$
with
result: $result2 \subseteq result$
then
act2: $last\_endpoint\_active\_query\_result := result2$
end
Event query_health_apis ⟨ordinary⟩ $\widehat{=}$
extends query_health_apis
any
$an\_endpoint$
$result$
where
grd1: $an\_endpoint \in E$
grd2: $result \subseteq HCA$
grd3: $an\_endpoint \in union(ran(endpoints)) \Rightarrow result = \{health\_apis(an\_endpoint)\}$
grd4: $an\_endpoint \notin union(ran(endpoints)) \Rightarrow result = \varnothing$
then

act1: $last\_health\_query\_result := result$

end

Event clock ⟨ordinary⟩ $\widehat{=}$

any

health_api_responses

where

grd1: $health\_api\_responses \in union(ran(endpoints)) \rightarrow BOOL$

then

act1: $time := time + 1$

act2: $unavailable\_endpoints := \{x | x \in dom(health\_api\_responses) \wedge health\_api\_responses(x) = FALSE\}$

end

END