UML state machines

Matt

Learning hierarchical and concurrent state machines

Bunch

Bunch is a tool to cluster states

Kirill E Bogdanov

16:16

DFASAT

Kirill E Bogdanov

map state to a 'cluster state'

and 'cluster state' aggregate all outgoing transitions

when you add a new state to a collection, you add its outgoing transitions (if needed) to the transitions of the 'cluster state'

every time a 'cluster state' changes, you need to look at outgoing transitions and make sure you know that you need to merge targets of those states

This means that every time you add a state to a cluster

you need to add target states for transitions from this state

to clusters associated to states from the ealier version of the cluster state

assume we have:

A-a->B,A-b->C

A is a cluster with some states

same for B and C

if you are adding D-a->E

it means that you need to add D to the A cluster

and E needs adding to the B cluster

and if D and E are also clusters

all the corresponding elements need updating

One way to represent clusters is 'union-find'

which makes it possible to efficiently merge sets

and find a 'representative state'

'representative state' (top of the tree for union-find)

will be storing all the transitions

all the internal states may have transitiosn

but it does not matter - you duplicate them if needed from the representative state

hence when merge clusters A and D

you look at outgoimg transtiions from the representative states of those clusters

and merge those clusters recurvively

where 'merge' means that you connect 'representative states'

and copy transitions from the representative state of D one to the representative state of A

(copy if needed)

In the example I had, D had one transition

and A had two

'a' transition from D would go to E

so when merging A and D you would need to add 'a' if not already from representative state of A

and it there is one, you need to merge clusters that 'a' points to from A and that of E

For each cluster, you would need to record what needs doing, because whilst merging A and D

you would need to merge B and E and so on,

eventually you might find yourself recursing to merge A and D again.

You need to have a worklist

because merging A and D requires merging B and E and there could be transition on 'b' from both A and D that also requires merging

Hence once you start merging A and D, you record this hence if you recurse, you would know you already doing it

but for transitions 'a', 'b' etc you would need a worklist to indicate what you merged and what you have not yet merged.

For each cluster you could have a set of labels that need handling. Initially, all of the labels and once you merge one, you recurse (or not - you only need to record what needs merging in the target clusters)

Take a representative state, record which labels and clusters need merging

then you can process this one cluster at a time, revisiting old clusters if new states are added to them.

You

if we adding

F -b-> G

F will be merged to A,D cluster and G will merged with C

here,we

1. given a state, which group does it belong to

2. given a group, which transitions are there from any states in this group

given a group, which labels have been processed

3. given a group, which labels have been processed

Make sure you write tests to check that your code works

Develop examples and expected merge results

make sure your tests run on those examples and compared obtained results with expected results.