4-02-16

EX1

Alphabet:

S(x), SM(x), C(x), Sh(x), H(x), incommand(x,y), onboard(x,y), been(x,y)

Axioms:

Forall x. SM(x) implies S(x) \\ ISA

Forall x. C(x) implies S(x) \\ ISA

Forall x. C(x) implies not SM(x) \\ disjoint

Forall x. S(x) implies C(x) or SM(x) \\ complete

Forall x,y. incommand(x,y) implies C(x) and Sh(y) \\ typing

Forall x,y. incommand(x,y) implies onboard(x,y) \\ subset

Forall x. C(x) implies 1<=#{y|incommand(x,y)}<=10\\ multi after refinement

Forall y. Sh(y) implies #{x|incommand(x,y)} <=1 \\ multi

Forall x,y. been(x,y) implies Sh(x) and H(y) \\ typing

Forall x,y. onboard(x,y) implies S(x) and Sh(y) \\ typing

Forall x. S(x) implies 1<=#{y|onboard(x,y)}<=10 \\ multi after refinement

EX2

We need to complete this instantiation of this UML diagram T. We do it, following this procedure

Iold =0, Inew = I

While (Inew and Iold are different) do

Iold= Inew

For each (forall x. A(x) -> B(x) in T) do

For each a in A^Inew do

B^Inew = B^Inew union{a}

Similar for each subset constraints forall x,y. P(x,y) -> R(x,y) in T

I = Inew

Return I

I = (Obj^I, S^I,SM^I,C^I,onboard^I,incommand^I,Sh^I,H^I,been^I)

I0:

S^I={}

SM^I={Dustin,Lubber,Rusty}

C^I={Alice,Helen}

onboard^I={ (Dustin,Constitution),(Dustin,Bumpy),(Rusty,Bumpy )}

incommand^I={ (Alice,Constitution),(Helen,Enterprise )}

Sh^I={ Bumpy,Lumpy,Constitution,Enterprise }

H^I={ Genoa,Calais,Piraeus }

been^I ={ (Constitution,Genoa),(Constitution,Calais),(Bumpy,Calais)}

I1:

S^I={ Dustin,Lubber,Rusty , Alice,Helen }

SM^I={Dustin,Lubber,Rusty}

C^I={Alice,Helen}

onboard^I={ (Dustin,Constitution),(Dustin,Bumpy),(Rusty,Bumpy ), (Alice,Constitution),(Helen,Enterprise )}

incommand^I={ (Alice,Constitution),(Helen,Enterprise )}

Sh^I={ Bumpy,Lumpy,Constitution,Enterprise }

H^I={ Genoa,Calais,Piraeus }

been^I ={ (Constitution,Genoa),(Constitution,Calais),(Bumpy,Calais)}

I2=I1

The instantiation is complete

Now we need to check if I2|=T where T is UML diagram, this means that we want to check if all axioms in T evaluate to true in the interpretation I2. The instantiation is correct.

1. q(x)<- Sh(x) and Exists y,y’. onboard(y,x) and S(y’) and onboard(y’,x) and y’=y

q(x):{Constitution,Bumpy}

1. q(x)<- S(x) and forall y. not onboard(x,y) implies Sh(y)

q(x) :{ Lubber}

1. q() <- Exists x. Sh(x) and (forall y. onboard(x,y) implies (S(x) and not C(x))

q(): true

EX3

Model checking a closed mu calculus formula phi over transition system T = <S,Ra,Pi> (S is set of states, Ra set of transitions and Pi mapping function from a set of propositions P to a subset of S) means that we want to verify that the initial state of T is in the extension of phi over T. Model checking return a subset of S in which each state satisfied phi. To compute this subset we need to use the labelling algorithm, that consist in labelling the states of T with predicates that are true in them. The extensions of least fixpoint and greatest fixpoint are computed by applying the Tarksi-Knaster approximates theorem.

vX.muY.((a or b) and [next]X) or ([next]Y)

We are going to find the greatest fixpoint (GFP) because of the presence of vX

[X0] = {0,1,2,3,4}

[X1] = [muY.((a or b) and [next]X0) or ([next]Y)]

We are going to find the least fixpoint (LFP) because of the presence of muY

[Y00] = {}

[Y01]=[((a or b) and [next]X0) or ([next]Y00)] = [([a] union [b]) intersect PreA(next,[X0])) union PreA(next,[Y00]) = ({0,1} union {4}) intersect {0,1,2,3,4}) union {} = {0,1,4}

[Y02]=[((a or b) and [next]X0) or ([next]Y01)] = [([a] union [b]) intersect PreA(next,[X0])) union PreA(next,[Y01]) = ({0,1} union {4}) intersect {0,1,2,3,4}) union {0,3,4} = {0,1,34}

[Y03]=[((a or b) and [next]X0) or ([next]Y02)] = [([a] union [b]) intersect PreA(next,[X0])) union PreA(next,[Y02]) = ({0,1} union {4}) intersect {0,1,2,3,4}) union {0,3,4} = {0,1,34}

[Y03] = [Y02] - - > found LFP

[X1] = {0,1,3,4}

[X2] = [muY.((a or b) and [next]X1) or ([next]Y)]

We are going to find the least fixpoint (LFP) because of the presence of muY

[Y10] = {}

[Y11]=[((a or b) and [next]X1) or ([next]Y10)] = [([a] union [b]) intersect PreA(next,[X1])) union PreA(next,[Y10]) = ({0,1} union {4}) intersect {0,3,4}) union {} = {0,4}

[Y12]=[((a or b) and [next]X1) or ([next]Y11)] = [([a] union [b]) intersect PreA(next,[X1])) union PreA(next,[Y11]) = ({0,1} union {4}) intersect {0,3,4}) union {3,4} = {0,3,4}

[Y13]=[((a or b) and [next]X1) or ([next]Y12)] = [([a] union [b]) intersect PreA(next,[X1])) union PreA(next,[Y12]) = ({0,1} union {4}) intersect {0,3,4}) union {0,3,4} = {0,3,4}

[Y13] = [Y12] - - > found LFP

[X2] = {0,3,4}

[X3] = [muY.((a or b) and [next]X2) or ([next]Y)]

We are going to find the least fixpoint (LFP) because of the presence of muY

[Y20] = {}

[Y21]=[((a or b) and [next]X2) or ([next]Y20)] = [([a] union [b]) intersect PreA(next,[X1])) union PreA(next,[Y20]) = ({0,1} union {4}) intersect {0,3,4}) union {} = {0,4}

[Y22]=[((a or b) and [next]X2) or ([next]Y21)] = [([a] union [b]) intersect PreA(next,[X2])) union PreA(next,[Y21]) = ({0,1} union {4}) intersect {0,3,4}) union {3,4} = {0,3,4}

[Y23]=[((a or b) and [next]X2) or ([next]Y22)] = [([a] union [b]) intersect PreA(next,[X2])) union PreA(next,[Y22]) = ({0,1} union {4}) intersect {0,3,4}) union {0,3,4} = {0,3,4}

[Y23] = [Y22] - - > found LFP

[X3] = {0,3,4}

[X3] = [X2] - - > found GFP

It is 0 in {0,3,4} ? YES, phi satoisfied by transition system

Now we need to do model checking with CTL formula. Give a Kripke model (KM) = <S,I,R,AP,L> (S set of states, I set of initial states, R set of transitions, AP set of atomic propositions and L labelling function) and a CTL formula phi, we want to verify if KM,s |=phi where s is state of S. With model checking we return a subset of S in which each state satisfies phi. To compute it we need to exploit the syntactic structure of CTL formula, in particular we need to translate each CTL sub formula into mu calculus formula and then applying the labelling algorithm to find their extensions.

AF(EG(a implies AX EX (not a and not b)))

alpha = EX (not a and not b) = <next> (not a and not b)

beta = AX alpha

gamma = a implies beta

delta = EG gamma

sigma = AF delta

[alpha] = [<next> (not a and not b)] = PreE(next,[not a and not b]) = PreE(next, {2,3,4} intersect {0,2,3}) = PreE(next,{2,3}) = {0,1,2}

[beta] = [AX alpha] = [[next] alpha] = PreA(next,[alpha]) = {0,1,2,4}

[gamma] = [a implies beta] = [not a or beta] = [not a] union [beta] = {2,3,4} union {0,1,2,4} = {0,1,2,3,4}

[delta] = [EG gamma] = [vX gamma and <next>X]

We are going to find the greatest fixpoint (GFP) because of presence of vX

[X0] = {0,1,2,3,4}

[X1] = [gamma and <next> X0] = [gamma] intersect PreE(next,[X0]) = {0,1,2,3,4}

intersect {0,1,2,3,4} = {0,1,2,3,4}

[X1] = [X0] - - -> found GFP

[delta] = {0,1,2,3,4}

[sigma] = [AF delta] = [mu X delta or [next] X]

We are going to find the least fixpoint (LFP) because of presence of muX

[X0] = {}

[X1] = [delta or [next] X0]= [delta] union PreA(next,[X0]) = {0,1,2,3,4} union {} = {0,1,2,3,4}

[X2] =[delta or [next] X1]= [delta] union PreA(next,[X1]) = {0,1,2,3,4} union {0,1,2,3,4} = {0,1,2,3,4}

[X2] = [X1] - - >found LFP

[sigma] = {0,1,2,3,4}

It is 0 in [sigma] ? YES, so the CTL formula is true in this transition system

EX4

Two states of transition system are bisimular if they have the same behaviour. This means:

* Locally two states look undistinguishable
* Every action that can be done on one it is also done on the other one

A binary relation R is a bisimulation iff (s,t) in R implies that

* s is final iff t is final
* Forall action a:
  + If s-> s’ then exists t’. t->t’ and (s’,t’) in R
  + If t->t’ then exists s’. s->s’ and (s’,t’) in R

A state s0 of transition system S is bisimular to a state t0 of transition system T iff exists a bisimulation between initial states s0 and t0.

The algorithm used to compute bisimulation is the following:

1. R = SxT
2. R’ = R- {(s,t) such that not(s is final in S eq t final in T)}
3. While (R noteq R’)

R=R’

R’’ = R’ – {(s,t) such that exists s’,a. s action a (->) s’ and not exists t’. t action a (->) t’ and (s’,t’) in R’}

{(s,t) such that exists t’,a. t action a (->) t’ and not exists s’. s action a (->) s’ and (s’,t’) in R’}

We start with the first step:

1. Assuming that all state in S are equal to all states in T

R0 = {(t1,q1),(t1,q2),(t1,q3),(t1,q4),(t1,q5), (t2,q1),(t2,q2),(t2,q3),(t2,q4),(t2,q5)}

1. Now we remove all pairs that have one state that is final and other one that is not final

R1 = {(t1,q1),(t1,q4),(t1,q5), (t2,q2),(t2,q3)}

1. Now we repeat iteratively that we remove all pairs that one state can do an action and the other state cannot copy it.

R2 = {(t1,q1), (t2,q2),(t2,q3)}

* t1 can do a and go to t2 and q4 can do a and go to q5 but (t2,q5) is not in R1. So I remove (t1,q4)
* t1 can do a and go to t2 and q5 cannot copy it. So I remove (t1,q5)

R3 = {(t1,q1), (t2,q2)}

* t2 can do b and go to t1 and q3 can do b and go to q4 but (t1,q4) is not in R2 so I remove (t2,q3)

R4 = {(t2,q2)}

* t1 can do a and go to t2 and q1 can do a and go to q3 but (t1,q3) is not in R3 so I remove (t1,q1)

R5 = {}

* t2 can do b and go to t1 and q2 can do b and go to q1 but (t1,q1) is not in R4 so I remove it.

R6 = {}

R6 = R5

S and T are not bisimular since (t1,q1) is not in R6

EX5

q(x)<-contains(x,y),genre(y,z)

Possible x that can resolve this query are cd1,null2,cd2, cd3,cd4,null4

But we cannot return null values so the result of this query are {cd1,cd2,cd3,cd4}

q(x,z)<-contains(x,y),genre(y,z)

In this other case we want to return 2 values, so the possible couples are:

(cd1,rock),(cd2,rock),(null2,rock),(cd3,rock),(cd4,null5),(null4,null5) but we cannot return null values so the real results are:

{(cd1,rock),(cd2,rock), (cd3,rock)}