16-12-15

EX1

Alphabet:

S(x), B(x), H(x),Sh(x), TB(x), L(x), onboard(x,y), incommand(x,y), been(x,y) ,carried(x,y)

Axioms:

Forall x. Sh(x) implies B(x) \\ ISA

Forall x. TB(x) implies B(x) \\ ISA

Forall x. Sh(x) implies TB(x) \\ disjoint

Forall x. B(x) implies Sh(x) or TB(x) \\ complete

Forall x,y. carried(x,y) implies Sh(x) and L(y) \\ typing

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

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

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

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

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

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

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

EX2

1. We need to complete this instantiation, so we are going to follow this procedure give a UML diagram T:

Iold=0, Inew = I

While (Inew and Iold are different) do

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,B^I,H^I,Sh^I,TB^I,L^I,onboard^I,incommand^I,been^I,carried^I)

I0:

S^I = {Dustin,Lubber,Rusty}

B^I = {}

H^I = {Napoli, Calais, Piraeus}

Sh^I = {Constitution}

TB^I = {Bumpy, Lumpy}

L^I = {}

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

incommand^I = {(Lubber,Constitution)}

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

carried^I = {}

I1:

S^I = {Dustin,Lubber,Rusty}

B^I = {Constitution, Bumpy, Lumpy }

H^I = {Napoli, Calais, Piraeus}

Sh^I = {Constitution}

TB^I = {Bumpy, Lumpy}

L^I = {}

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

incommand^I = {(Lubber,Constitution)}

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

carried^I = {}

I2:

S^I = {Dustin,Lubber,Rusty}

B^I = {Constitution, Bumpy, Lumpy }

H^I = {Napoli, Calais, Piraeus}

Sh^I = {Constitution}

TB^I = {Bumpy, Lumpy}

L^I = {}

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

incommand^I = {(Lubber,Constitution)}

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

carried^I = {}

I2 = I1 the instantiation is complete

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

Ok Foreach instance of Ship there is one and only one instance of Sailor in the association incommand. It is verified. Each elements of Ship and TugBoat are also elements of Boat, so ok. Each pairs in incommand are also in onboard associations, so fine. The other this are ok, so the instantiation is correct.

1. q(x)<- S(x) and Exists y. onboard(x,y) and B(y) and been(y, Piraeus) and H(Piraeus)

q(x): { Constitution }

1. q() <- Exists x. H(x) and Exists y,y’. B(y,x) and B(y’,x) and y’ noteq y

q() <-true

1. q(x) <- B(x) and forall y. been(x,y) implies H(y)

q(x): { Constitution}

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.

νX.muY.((a and <next> X) or ([next]Y))

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

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

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

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

[Y00] = {}

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

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

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

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

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

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

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

[Y10] = {}

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

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

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

[Y14] = [(a and <next> X1) or ([next]Y13)] = ([a] intersect PreE(next, [X1]) union PreA(next,[Y13]) = ({0,1,4} intersect {0,1,3,4}) union {0,3,4} = {0,1,3,4}

[Y14] = [Y13] - - > found LFP

[X2] = {0,1,3,4}

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

It is 0 in [phi] = {0,1,3,4}? YES, so phi is satisfied by this transition system

Now we want to do model check with CTL formula. Given a Kripke model KM = <S,I,R,AP,L> (S set of states, I set of initial states, R is set of transition systems, Ap set of atomic propositions, L labelling function L:S->2^AP) and CTL formula phi we want to verify that 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 translate CTL sub formula into mu calculus formula and then we apply labelling algorithm to them to find their extensions.

EF (AG (a implies AX not a))

alpha = AX not a = [next] not a

beta = a implies alpha = not a or [next] not a

gamma = AG beta

delta = EF gamma

[alpha] = [[next] not a] = PreA(next,[not a]) =PreA(next,{2,3}) = {0,1,2}

[beta] = [not a or alpha] = [not a] union [alpha] = {2,3} union {0,1,2} = {0,1,2,3}

[gamma] = [AG beta] = [vX beta and [next] X]

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

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

[X1] = [beta and [next] X0] = [beta] intersect PreA(next,[X0]) = {0,1,2,3} intersect {0,1,2,3,4} = {0,1,2,3}

[X2] =[beta and [next] X1] = [beta] intersect PreA(next,[X1]) = {0,1,2,3} intersect {0,1,2,4} = {0,1,2}

[X3] =[beta and [next] X2] = [beta] intersect PreA(next,[X2]) = {0,1,2,3} intersect {1,2,4} = {1,2}

[X4] =[beta and [next] X3] = [beta] intersect PreA(next,[X3]) = {0,1,2,3} intersect {1,2,4} = {1,2}

[X4] = [X3] - - > found GFP

[gamma] = {1,2}

[delta] = [EF gamma] = [muX gamma or <next> X]

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

[X0] = {}

[X1] = [gamma or <next> X0] = [gamma] union PreE(next,[X0]) = {1,2} union {} = {1,2}

[X2] = [gamma or <next> X1] = [gamma] union PreE(next,[X1]) = {1,2} union {0,1,2} = {0,1,2}

[X3] = [gamma or <next> X2] = [gamma] union PreE(next,[X2]) = {1,2} union {0,1,2,4} = {0,1,2,4}

[X4] = [gamma or <next> X3] = [gamma] union PreE(next,[X3]) = {1,2} union {0,1,2,3,4} = {0,1,2,3,4}

[X5] = [gamma or <next> X4] = [gamma] union PreE(next,[X4]) = {1,2} union {0,1,2,3,4} = {0,1,2,3,4}

[X5] = [X4] - - >found LFP

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

It is 0 in {0,1,2,3,4} ? YES, 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 of them can also be done on the other

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 T iff there exist a bisimulation between initial state s0 and t0.

Following bisimulation algorithm we are going to see if S and T are bisimuar

1. Assuming that every state of S is equivalent to every state of T

R0={(s1,t1),(s1,t2), (s2,t1),(s2,t2),(s3,t1),(s3,t2)}

1. Remove states that are final in one and not final in other

R1 = {(s1,t1), (s2,t2),(s3,t2)}

1. Now we enter in the loops and remove iteratively all pairs where one can do an action and the other one cannot copy the other.

R2 = {(s1,t1), (s2,t2),(s3,t2)}

R2 = R1

S an T are bisimular

EX5

q1() :- e(a,y),e(x,y),e(x,b)

q2() :- e(a,y),e(x,y),e(x,z),e(w,z),e(w,b)

we need to check if q1 is contained in q2, so this means that we want to check if q1 (x) implies q2(x) is valid. Valid means that forall I,alpha |= forall x. q1(x) implies q2(x)

We know that in FOL the validity is undecidable but with conjunctive queries we can make it satisfiable because we can transform queries in databases.

We need three steps:

1. Freeze variable, i.e substitute free variable with fresh variables

In this case we don’t need this operation because q1 and q2 are already Boolean

1. Build canonical database correspond of q1. Remenber that we can solve q1() is subseteq q2 () if we can extract the database of q1 and check on it the query q2

Iq1 = (delta^I1,e^I1,c^I1)

delta^I1={a,x,y,b}

e^I1 = {(a,y), (x,y), (x,b)}

c^I1 = {}

1. Check if q2 is true over database of q1, so verify if there is an assignment for all free variables.

alpha(a) = a

alpha(y) = y

alpha(x) = x

alpha(z) = b

alpha(w) = x

alpha(b) = y

This is a satisfying assignment

Now we are going to find an homomorphism. Homomorphism is a mapping between two interpretation, between elements of 2 domains: h:delta^I ->delta^J such that:

* h(c^I) = c^J
* (x,y) in e^I then (h(x),h(y)) in e^J

Where e is predicate

Find a homomorphism means that we want to guess a mapping and show that respect two properties above. Remember that there is a theorem that says that if you have an assignment alpha, that is a satisfying assignment, you can transform alpha in 2 homomorphism between two canonical representation, Iq1|=q2 iff h:Iq1|=Iq2

Iq2 = (delta^I2,e^I2,c^I2)

delta^I2={a,y,x,z,w,b}

e^I2= { (a,y),(x,y),(x,z),(w,z),(w,b)}

c^I2 = {}

Now we need to understand if two properties are satisfied. For constant property is satisfied, we need to check the second property

(a,y) in e^I2 then (h(a),h(y)) in e^I1 ok because (a,y) is in e^I1

(x,y) in e^I2 then (h(x),h(y)) in e^I1 ok because (x,y) is in e^I1

(x,z) in e^I2 then (h(x),h(z)) in e^I1 ok because (x,b) is in e^I1

(w,z) in e^I2 then (h(w),h(z)) in e^I1 ok because (x,b) is in e^I1

(w,b) in e^I2 then (h(w),h(b)) in e^I1 ok because (x,b) is in e^I1