12-01-2016

EX1

Alphabet:

S(x), C(x), Sh(x), TB(x), B(x), H(x), onboard(x,y), incommand(x,y), been(x,y), worksin(x,y)

Axioms:

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

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

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

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

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

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

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

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

Forall x. Sh(x) implies not TB(x) \\ disjoint

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

Forall x,y. worksin(x,y) implies TB(x) and H(y) \\ typing

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

Forall x,y. worksin(x,y) implies been(x,y) \\ subset

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

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

EX2

1. We need to complete this instantiation because we have some ISA between classe in UML diagram T. We are going to follow this procedure:

Iold = 0, Inew = =

While(Iold and Inew 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)

I= Inew

Return I

I= (Obj^I, S^I,C^I,Sh^I,TB^I,incommand^I,onboard^I,B^I,been^I,H^I,worksin^I)

I0:

S^I = {Dustin,Rubber,Rusty}

C^I = {Alice,Jim}

Sh^I = {Constitution, Enterprise}

TB^I = {Bumpy, Lumpy}

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

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

B^I = {}

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

H^I = {Genoa,Calais, Piraeus}

worksin^I = {(Bumpy,Calais),(Lumpy,Calais)}

I1:

S^I = {Dustin,Rubber,Rusty, Alice,Jim }

C^I = {Alice,Jim}

Sh^I = {Constitution, Enterprise}

TB^I = {Bumpy, Lumpy}

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

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

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

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

H^I = {Genoa,Calais, Piraeus}

worksin^I = {(Bumpy,Calais),(Lumpy,Calais)}

I2 = I1 so the instantiation is complete

Now we need to check if I2 |= T, this means that we want to verify if all axioms are true in this interpretation. And in this case it is correct

1. q(x)<-S(x) and Exists y. onboard(x,y) and B(y) and Exists z. been(y,z) and H(z) and Exists k. worksin(k,z) and TB(k)

q(x) :{Rusty}

1. q()<- Exists. x H(x) and Exits y,y’. been(y,x) and B(y) and TB(y) and been(y’,x) and B(y’) and TB(y’)
2. q(x) <- S(x) and forall y. been(x,y) implies H(y)

EX3

Model checking a closed mu calculus formula phi over transition system T = <S,Ra,Pi> (S is set of states, Ra is set of transitions, Pi is mapping function from a set of propositions P to a subset of S) means that we want to verify if the initial state of T is in the extension of phi over T. With model checking we return a subset of S in which each state satisfies phi. To compute it we need to apply labelling algorithm that consists in labelling states of T with predicates that are true in them. The extensions of least fixpoint and greatest fixpoint are computed using Tarski-Knaster algorithm.

νX.muY.((a and [next]X) or ([next]Y))

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

[Y00 ] = {}

[Y01] =[(a and [next] X0) or [next] Y00]= ([a] intersect PreA(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 PreA(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 PreA(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 the least fixpoint (LFP) because of presence of muY

[Y10 ] = {}

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

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

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

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

[X2] = {0,3,4}

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

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

[Y20 ] = {}

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

[Y22] =[(a and [next] X2) or [next] Y21]= ([a] intersect PreA(next,[X2])) union PreA(next,[Y11]) = ({0,1,4} intersect {0,3,4}) union {3,4} = {0,3,4}

[Y23] =[(a and [next] X2) or [next] Y22]= ([a] intersect PreA(next,[X2])) union PreA(next,[Y12]) = ({0,1,4} 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, so phi is satisfied by this transition system

Now we do model checking with CTL formula. Given a Kripke model (KM) = <S,I,R,AP,L> (S is set of states, I is set of initial states, R is set of transitions, AP set of atomic propositions and L is labelling function L:S->2^AP) and a CTL formula phi, model checking means that KM, s |= phi where s is state of S. With model checking we return a subset of states of S in which each state satisfy phi. To compute this subset we need to exploit syntactic structure of CTL formula, in particular, we translate each sub formula of CTL in mu calculus formula and we apply to them the labelling algorithm to find their extensions.

EF (AG (a implies EX EX not a))

alpha = EX not a = <next> not a

beta = EX alpha = <next><next> not a

gamma = a implies beta

delta = AG gamma

sigma = EF delta

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

[beta] = [<next> alpha] = PreE(next,[alpha]) = PreE(next, {0,1,2}) = {0,1,2,4}

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

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

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

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

[X1] = [ gamma and [next] X0] = [gamma] intersect PreA(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] = [EF delta] = [mu X delta or <next> X]

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

[X0] = {}

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

[X2] = [delta or <next> X1] = [delta] union PreE(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

Check if the Hoare triple {P} while g do S {Q} is correct cannot be done automatically, we need to use inference rule.

We assume to have a candidate invariant I, and we need to show

1. P implies I
2. {I and g} S {I} = {I and g} implies wp(S,I)
3. {I and not g) implies Q

If the candidate I satisfies these 3 properties it is called invariant.

If the invariant is give these operations are computed in automatic otherwise it is very difficult to work with it. The problem of checking if a triple is correct or not is undecidable, there is not a sound or complete technique to do this or to generate all possible invariants. We can only have a sound technique to decide if that candidate is invariant or not.

Invariant : (0<=i and 0<=j and i+j <= 5).

{i=0 AND j=5} while(i<5) do (j=j-1; i:= i+1) {j=0}

* P implies I

{i=0 and j = 5} implies {0<=i and 0<=j and i+j <=5}

It is correct because if i = 0 and j = 5, i and j are major and equal than 0 and also i+j is <=5

* {I and not g} implies Q

{0<=i and 0<=j and i+j <=5 and i>=5} implies {j=0}

{0<=i and i+j <=5 and i>=5} implies {j=0}

It is ok because if i is major e and equal of 5 j it should be equal to 0 to have that i+j <=5

* {I and g} S {I}

We need to compute the weakest precondition, wp(S,I)

{i>=1 and j-1>=0 and i+j-1<=4} = {i>=1 and j>=1 and i+j<=5} is wp

{j=j-1}

{i+1>=0 and j>=0 and i+1+j<=5}= {i>=1 and j>=0 and i+j<=4}

{i=i+1}

{0<=i and 0<=j and i+j <=5}

{I and g} implies wp(S,I)

{0<=i and 0<=j and i+j <=5 and i<5} implies {i>=1 and j>=1 and i+j<=5}

{0<=i<5 and 0<=j and i+j <=5} implies {i>=1 and j>=1 and i+j<=5}

It is not true because i could be equal to 0 as j so the condition is not correct

{0<=i and 0<=j and i+j <=5} is not an invariant but we cannot say anything about the hoare triple, if it is correct or not

EX5

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

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

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,y}

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

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(z) = a

alpha(w) = 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,z,w}

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

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

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

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

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

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

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

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