09-07-2015

EX1

Alphabet:

E(x), W(x), D(x), TD(x), worksIn(x,y), directs(x,y), Dep(x), Pa(x),Pr(x), pd(x,y), pp(x,y), pr(x,y), R(x)

Axioms:

Forall x. TD(x) implies W(x) \\ ISA

Forall x. TD(x) implies D(x) \\ ISA

Forall x. W(x) implies E(x) \\ ISA

Forall x. D(x) implies E(x) \\ ISA

Forall x,y. directs(x,y) implies D(x) and Dep(y) \\ typing

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

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

Forall x,y. worksIn(x,y) implies E(x) and Dep(y) \\ typing

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

Forall y. Dep(y) implies 1<=#{x|worksIn(x,y)} \\ multi after refinements

Forall x,y. directs(x,y) implies worksIn(x,y) \\ subset

Forall x,y. pd(x,y) implies Dep(x) and Pa(y) \\ typing

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

Forall x,y. pr(x,y) implies R(x) and Pa(y) \\ typing

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

Forall x,y. pp(x,y) implies Pr(x) and Pa(y) \\ typing

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

Forall x,x’,y. pd(x,y) and pp(x’,y) implies x = x’ \\ Key

EX2

1. To complete the instantiation of UML diagram T we need to consider this procedure:

Iold=0, Inew = I

While (Inew and Iold are different) do

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

Foreach a in A^Inew do

B^Inew = B^Inew union {a}

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

I=Inew

Return I

I= (Obj^I, E^I,W^I,D^I,TD^I, worksIn^I, directs^I, Dep^I, pd^I,pr^I,pp^I,Pa^I,Pr^I,R^I)

I0:

E^I = {}

W^I = {John, Mary,Joe}

D^I={Jim,Rick}

TD^I={Ann}

worksIn^I={(John,ICT),(Mary,ICT),(Joe,HR) }

directs^I= {(Ann,ICT), (Jim,HR),(Rick, Management)}

Dep^I={ICT,HR,Management}

pd^I={}

pr^I={}

pp^I={}

Pa^I={}

Pr^I={}

R^I={}

I1:

E^I = {}

W^I = {John, Mary,Joe,Ann}

D^I={Jim,Rick,Ann}

TD^I={Ann}

worksIn^I={(John,ICT),(Mary,ICT),(Joe,HR)}

directs^I= {(Ann,ICT), (Jim,HR),(Rick, Management)}

Dep^I={ICT,HR,Management}

pd^I={}

pr^I={}

pp^I={}

Pa^I={}

Pr^I={}

R^I={}

I2:

E^I = {John, Mary,Joe,Jim,Rick,Ann}

W^I = {John, Mary,Joe,Ann}

D^I={Jim,Rick,Ann}

TD^I={Ann}

worksIn^I={(John,ICT),(Mary,ICT),(Joe,HR), (Ann,ICT), (Jim,HR),(Rick, Management)}

directs^I= {(Ann,ICT), (Jim,HR),(Rick, Management)}

Dep^I={ICT,HR,Management}

pd^I={}

pr^I={}

pp^I={}

Pa^I={}

Pr^I={}

R^I={}

I3=I2 so the instantiation is complete

Now we want to check if it is correct, so I3 |= T where T is the UML diagram, this means that we need to verify if all axioms are true in the interpretation I3

I3 is correct because every axioms result correct. In E there are all instance of its subclasses, so instance of W and D. In W there also instance of TD and the same is for D. To each instance of D is associate al most one Dep for the association directs, and for a Dep there is one and only one D that directs it. Each instance of E has a association worksIn with one and only one Dep.

1. q(x) <-D(x) and Exists y. Dep(y) and directs(x,y) and Exists z. W(z) and worksIn(x,y)

q(x) : {Jim,Rick,Ann}

1. q(x) <- Dep(x) and (Forall y,z. (E(y) and D(z) and worksIn(y,x)) implies y=z)

q(x) : {Management}

EX3

Model checking a closed mu calculus formula phi over transition system T = <S,Ra,Pi> (S is set of state, 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 this subset we apply the labelling algorithm, that consist in labelling states of T with predicates that are true in them. The extensions of least fixpoint and greatest fixpoint are computed using the Taski-Knaster approximates theorem.

νX.muY.((a and <next> X) or (not b and <next> Y))

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

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

[X1] = [mu Y (a and <next>X0) or (not b and <next>Y)]

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

[Y00] = {}

[Y01] = [(a and <next>X0) or (not b and <next>Y00)] = ([a] intersect PreE(next,[X0])) union ([not b] intersect PreE(next,[Y00])) = ({2,4} intersect {1,2,3,4}) union ({1,2} intersect {}) = {2,4} union {} = {2,4}

[Y02] = [(a and <next>X0) or (not b and <next>Y01)] = ([a] intersect PreE(next,[X0])) union ([not b] intersect PreE(next,[Y01])) = ({2,4} intersect {1,2,3,4}) union ({1,2} intersect {1,2,3,4}) = {2,4} union {1,2} = {1,2,4}

[Y03] = [(a and <next>X0) or (not b and <next>Y02)] = ([a] intersect PreE(next,[X0])) union ([not b] intersect PreE(next,[Y02])) = ({2,4} intersect {1,2,3,4}) union ({1,2} intersect {1,2,3,4}) = {2,4} union {1,2} = {1,2,4}

[Y03] = [Y02] - - >{1,2,4} is LFP

[X1] = {1,2,4}

[X2] = [mu Y (a and <next>X1) or (not b and <next>Y)]

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

[Y10] = {}

[Y11] = [(a and <next>X1) or (not b and <next>Y10)] = ([a] intersect PreE(next,[X1])) union ([not b] intersect PreE(next,[Y10])) = ({2,4} intersect {1,2,3,4}) union ({1,2} intersect {}) = {2,4} union {} = {2,4}

[Y12] = [(a and <next>X1) or (not b and <next>Y11)] = ([a] intersect PreE(next,[X1])) union ([not b] intersect PreE(next,[Y11])) = ({2,4} intersect {1,2,3,4}) union ({1,2} intersect {1,2,3,4}) = {2,4} union {1,2} = {1,2,4}

[Y13] = [(a and <next>X1) or (not b and <next>Y12)] = ([a] intersect PreE(next,[X1])) union ([not b] intersect PreE(next,[Y12])) = ({2,4} intersect {1,2,3,4}) union ({1,2} intersect {1,2,3,4}) = {2,4} union {1,2} = {1,2,4}

[Y13] = [Y12] - - > {1,2,4} is LFP

[X2] = {1,2,4}

[X2] = [X1] - - > {1,2,4} is GFP

It is 1 in [phi] = {1,2,4}? YES, so the formula is satisfied by this transition system

Now we need to do model checking of 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 and AP is set of atomic propositions and L is labelling function L:S->2^AP) and CTL formula phi we want to verify KM,s|=phi where s is a state of S. Model checking return a subset of S, in which each formula satisfies phi. To compute it we need to exploit the syntactic structure of CTL formula, so are going to translate each sub formula of CTl into a mu calculus formula and then applying labelling algorithm to find their extensions.

(AG EFa )or E (aUb)

alpha = E(a U b) = mu X b or (a and <next> X)

beta =EF a = mu X a or <next> X

gamma= AG beta

delta= gamma or alpha

[alpha] =[ mu X b or (a and <next> X)]

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

[X0] = {}

[X1] = [b or (a and <next> X0)] = [b] union ([a] intersect PreE(next,[X0])) = {3,4} union ({2,4} intersect {}) = {3,4}

[X2] = [b or (a and <next> X1)] = [b] union ([a] intersect PreE(next,[X1])) = {3,4} union ({2,4} intersect {2,3,4}) = {3,4} union {2,4} = {2,3,4}

[X3] = [b or (a and <next> X1)] = [b] union ([a] intersect PreE(next,[X2])) = {3,4} union ({2,4} intersect {1,2,3,4}) = {3,4} union {2,4} = {2,3,4}

[X3] = [X2] so we have found the LFP

[alpha] = {2,3,4]

[beta] = [mu X a or <next> X]

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

[X0] = {}

[X1] = [a or <next> X0] = [a] union PreE(next,[X0]) = {2,4} union {} = {2,4}

[X2] = [a or <next> X1] = [a] union PreE(next,[X1]) = {2,4} union {1,2,3,4} = {1,2,3,4}

[X3] = [a or <next> X2] = [a] union PreE(next,[X2]) = {2,4} union {1,2,3,4} = {1,2,3,4}

[X3] = [X2] we found LFP

[beta] = {1,2,3,4}

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

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

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

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

[X1] = [X0] we foung GFP

[delta] = [gamma or alpha] = {1,2,3,4} union {2,3,4} = {1,2,3,4}

It is 1 in [delta] = {1,2,3,4}? YES, so the CTL formula is true in this transition system

EX4

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

We assume to have a candidate invariant I and we want 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 candidate I satisfies all these properties it is called invariant.

If we have a candidate this operations are automatically, otherwise it is very difficult to manage with. The problem of checking if a triple with while is true or not is undecidable because there no sound or complete technique to do this or to generate all possible invariants for the problems. We have only sound technique to check whether a candidate is invariant or not. If it not the case we cannot say anything about it.

{i=0} while(i<10) do i:= i+1 {i=10}

Candidate: i<=10

Check properties

* P implies I

{i=0} implies i<=10

Yes, because i=0 is always minor equal of 10

* (I and not g) implies Q

(i>10 and i<=10) implies i = 10

Yes

* The second property is a little bit difficult respect to the other, we need to compute the weakest precondition wp(S,I)

{i+1 <=10} = {i<=9} is the wp

i =i+1

{i=10}

i <= 10

Now we compute

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

(i<= 10 and i< 10) implies (i=9)

(i<10 ) implies (i<=9)

Yes it is satisfies because if I assume a value minor to 10 I should be always minor or equal to 9

EX5

q1() :- edge(r,b), edge(b,g), edge(g,r).

q2() :- edge(x,y), edge(y,z), edge(z,x), edge(z,v), edge(v,w), edge(w,z).

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={r,b,g}

e^I1 = {(r,b), (b,g), (g,r)}

c^I1 = {}

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

alpha(x) = r

alpha(y) = b

alpha(z) = g

alpha(v) = r

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={x,y,z,v}

e^I2= {(x,y),(y,z),(z,x),(z,v),(v,y)}

c^I2 = {}

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

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

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

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

(z,v) in e^I2 then (h(z),h(v)) in e^I1 ok because (g,r) is in e^I1

(v,y) in e^I2 then (h(v),h(y)) in e^I1 ok because (r,b) is in e^I1