13-02-2017

EX1

Alphabet:

S(x), Co(x), takes(x,y), credit(x,y), mark(x,y,z), Int(x), Cl(x), L(x)

Axioms:

Forall x. Cl(x) implies Co(x) \\ ISA

Forall x. L(x) implies Co(x) \\ ISA

Forall x. Cl(x) implies not L(x) \\ disjoint

Forall Co(x) implies Cl(x) or L(x) \\ complete

Forall x,y. credit(x,y) implies S(x) and Cl(y) \\ typing

Forall x,y,z. mark(x,y,z) implies credit(x,y) and Int(z) \\ typing

Forall x,y. credit(x,y) implies 1<=#{z|mark(x,y,z)}<=1 \\ multi

Forall x. S(x) implies 1<=#{y|credit(x,y)}<=30 \\ multi after refinement

Forall x,y. credit(x,y) implies takes(x,y) \\ subset

Forall x,y. takes(x,y) implies S(x) and Co(y) \\ typing

Forall x. S(x) implies 1<=#{y|takes(x,y)}<=30 \\ multi after refinement

EX2

1. The instantiation is not complete because of presence of ISA so we need to use following procedure to find a complete instantiation given a UML diagram T.

Iold = 0, Inew = I  
while(Iold and Inew are different) do

For each (forall x. A(x) implies 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) implies R(x,y)

I=Inew

Return I

I = (Obj^I,S^I,Co^I,Cl^I,L^I,credit^I,mark^I,takes^I)

I0:

S^I={peter,paul,mary,jane}

Co^I={}

Cl^I={calculus, AI,FM,algorithm}

L^I={IoT lab,db lab, hacking lab}

credit^I/mark^I={(peter,algorithm),(paul,calculus),(mary,algorithms,28),(mary,AI,30),(jane,FM,30),(jane,algorithms,30)}

takes^I = {(peter,IoT lab),(paul,IoT lab),(mary,FM),(jane,db lab),(jane,hacking lab),(jane,IoT lab)}

I1:

S^I={peter,paul,mary,jane}

Co^I={ calculus, AI,FM,algorithm, IoT lab,db lab, hacking lab }

Cl^I={calculus, AI,FM,algorithm}

L^I={IoT lab,db lab, hacking lab}

credit^I/mark^I={(peter,algorithm),(paul,calculus),(mary,algorithms,28),(mary,AI,30),(jane,FM,30),(jane,algorithms,30)}

takes^I = {(peter,IoT lab),(paul,IoT lab),(mary,FM), (jane,db lab),(jane,hacking lab), (jane,IoT lab), (peter,algorithm), (paul,calculus), (mary,algorithms), (mary,AI), (jane,FM), (jane,algorithms)}

I2 = I1 so the instantiation is complete

Now we need to check if the instantiation is correct, so if I2|=T, if all axioms of T are evaluate to true in the interpretation I2. It is correct.

1. A

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 find a greatest fixpoint GFP because of presence of vX

[X0] = {1,2,3,4,5}

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

We are going to find a 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]) = ({2,4,5} intersect {1,2,3,4,5}) union {} = {2,4,5}

[Y02] = [a and [next]X0) or [next]Y01] = ([a] intersect PreA(next,[X0]) union PreA(next,[Y01]) = ({2,4,5} intersect {1,2,3,4,5}) union {4,5} = {2,4,5}

[Y02] = [Y01] - - > found LFP

[X1] = {2,4,5}

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

We are going to find a 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]) = ({2,4,5} intersect {4,5}) union {} = {4,5}

[Y12] = [a and [next]X1) or [next]Y11] = ([a] intersect PreA(next,[X1])) union PreA(next,[Y11]) = ({2,4,5} intersect {4,5}) union {4,5} = {4,5}

[Y12] = [Y11] - - > found LFP

[X2] = {4,5}

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

We are going to find a least fixpoint LFP because of presence of muY

[Y20] = {}

[Y21] = [a and [next]X1) or [next]Y10] = ([a] intersect PreA(next,[X1])) union PreA(next,[Y10]) = ({2,4,5} intersect {4,5}) union {} = {4,5}

[Y22] = [a and [next]X1) or [next]Y11] = ([a] intersect PreA(next,[X1])) union PreA(next,[Y11]) = ({2,4,5} intersect {4,5}) union {} = {4,5}

[Y22] = [Y21] - - > found LFP

[X3] = {4,5}

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

It is [phi] = {4,5} ? NO, phi is satisfied by 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(not a implies (EX a and EX AG b))

alpha = AG b = vX b and <next> X

beta = EX alpha

gamma = EX a = <next> a = {2,4,5}

delta = gamma and beta

sigma = not a implies delta

tau = EF sigma

[alpha] = [vX b and <next> X]

We are going to find a greatest fixpoint GFP because of presence of vX

[X0] = {1,2,3,4,5}

[X1] = [b and <next> X0] = [b] intersect PreE(next,[X0]) = {3,4,5} intersect {1,2,3,4,5} = {3,4,5}

[X2] = [b and <next> X0] = [b] intersect PreE(next,[X0]) = {3,4,5} intersect {1,2,3,4,5} = {3,4,5}

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

[alpha] = {3,4,5}

[beta] = [EX alpha] = [<next> alpha] = PreE(next,[alpha]) = {1,2,3,4,5}

[delta] = [gamma and beta] = {2,4,5} intesect {1,2,3,4,5} = {2,4,5}

[sigma] = [not a implies delta] = [a or delta] = [a] union [delta] = {2,4,5} union {2,4,5} = {2,4,5}

[tau] = [EF sigma] = [mu X. sigma or <next> sigma]

We are going to find a least fixpoint LFP because of presence of mu X

[X0] = {}

[X1] = [sigma or <next> sigma] = [sigma] union PreE(next,[X0]) = {2,4,5} union {} = {2,4,5}

[X2] = [sigma or <next> sigma] = [sigma] union PreE(next,[X1]) = {2,4,5} union {1,2,3,4,5} = {1,2,3,4,5}

[X3] = [sigma or <next> sigma] = [sigma] union PreE(next,[X2]) = {2,4,5} union {1,2,3,4,5} = {1,2,3,4,5}

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

EX4

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

* Locally two states look undistinguishable
* Every action done on one it is also done 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 state t0 of transition system T iff there exist a bisimulation between state s0 and t0.

The algorithm to use to compute bisimulation is in the following

1. R=SxT
2. R’ = R-{(s,t) such that not(s is final in S eq t is 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 exist t’. t action a(->) t’ and (s’,t’) in R}

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

We start with first step:

1. Assuming that all state in S are equal to all state 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 one that is final and the other 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 can do one action and the other one cannot copy.

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

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 (t2,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 t1 but (t1,q1) is not in R4 so I remove (t2,q2)

R6 = {}

R6=R5

S and T are not bisimular because (t1,s1) is not in R6

EX5

q1(x) :- edge(x,y), edge(y,z), edge(z,x)

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

We want to check if q1 is contained in q2. So if forall x q1(x) implies q(2) is valid. Validity means that we want to verify that forall I,alpha |= forall x q1(x) implies q2(x) where I interpretation and alpha assignment.

In FOL validity is undecidable but with CQ we can make it satisfiable because transform query in database.

We need 3 steps:

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

q1(c) :- edge(c,y), edge(y,z), edge(z,c)

q2(c) :- edge(c,y), edge(c,w), edge(y,z), edge(z,c), edge(z,v), edge(v,y), edge(v,w), edge(w,z)

1. Build database corresponding of q1 because now q1 is “boolean” because I have fresh variable

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

delta^I1 = {c,y,z}

e^I1 = {(c,y),(y,z),(z,c)}

c^I1 = {c}

1. Check that q1(x) subseteq q2(x). So check if q2 is true over database of q1, verify if there is an assignment for all free variables

alpha(y) = y

alpha(w) = y

alpha(z) = z

alpha(v) = c

canonical database of q2:

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

delta^I2 = {c,y,z,w,v}

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

c^I2 = {c}

This is a satisfying assignment

Now we are going to show homomorphism. A homomorphism is a mapping between two interpretation, between elements of 2 domains h:delta^I implies delta^J such that:

1. h(c^I)=c^J
2. (x,y) in e^I then (h(x),h(y)) in e^J

Where e is the predicate

Find a homomorphism is to guess a mapping and show that these two properties are respected. Remember that there is a theorem that says that is you have an assignment, a satisfying assignment, you can transform it in two homomorphism between two canonical interpretation Iq1|=q2 iff Iq2 implies Iq1

The first property is satisfied. Now we need to check the second:

(c,y) in e^I2 then (h(c),h(y)) in e^I1 then (c,y) is in e^I1 so ok

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

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

(z,c) in e^I2 then (h(z),h(c)) in e^I1 then (z,c) is in e^I1 so ok

(z,v) in e^I2 then (h(z),h(v)) in e^I1 then (z,c) is in e^I1 so ok

(v,y) in e^I2 then (h(v),h(y)) in e^I1 then (c,y) is in e^I1 so ok

(v,w) in e^I2 then (h(v),h(w)) in e^I1 then (c,y) is in e^I1 so ok

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