18-02-2015

EX1

Alphabet:

P(x), host(x,y), numbr(x,y,z), A(x), M(x), Aq(x), C(x), lives(x,y), E(x), inhabits(x,y), We(x), Int(x)

Axioms:

Forall x,y. host(x,y) implies P(x) and A(y) \\ typing

Forall x. P(x) implies 10<=#{y|A(y)} \\ multi

Forall x,y,z. numbr(x,y,z) implies host(x,y) and Int(z) \\ typing

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

Forall x. M(x) implies A(x) \\ ISA

Forall x. Aq(x) implies A(x) \\ ISA

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

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

Forall x,y. inhabits(x,y) implies A(x) and E(y) \\ typing

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

Forall x,y. lives (x,y) implies inhabits(x,y) \\ subset

Forall x,y. lives(x,y) implies Aq(x) and We(y) \\ typing

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

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

EX2

1. To complete instantiation of UML diagram T we need to follow 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^new 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, Aq^I, C^I, We^I, lives^I)

I0:

Aq^I = {sawshark}

C^I = {dolphin,bluewhale}

We^I={ocean}

Lives^I = {(dolphin,ocean),(bluewhale, ocean),(sawshark,ocean)}

I1:

Aq^I = {sawshark, dolphin,bluewhale}

C^I = {dolphin,bluewhale}

We^I={ocean}

Lives^I = {(dolphin,ocean),(bluewhale, ocean),(sawshark,ocean)}

I2 = I1 so the instantition is complete

To verify if the instantiation I2 is correct we need to verify if I|=T, this means that checking if all axioms in T evaluate to true in the interpretation I

This interpretation is true because Aq contains all instances that are present in C. Each element in Aq has one and only instance of We in the association lives.

1. q(x,y)<- Animals(x) and Environment(y) and inhabits(x,y)
2. q(x)<- Mammals and (Forall y. Environment(y) implies inhabits(x,y) )

EX3

Model checking a closed mu calculus formula phi over transition system T = <S,Ra,Pi> (S set of states, Ra set of transitions, Pi a mapping function from a set of proposition P to a subset of S) means that we want to verify if 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 are going to apply the labelling algorithm, that it 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 Tarski-Knaster approximates theorem.

vXmuY((a and [next]X) or [next]Y)

we have first vX so we are going to compute the greatest fixpoint (GFP)

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

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

Now we have muY so we want to find the least fixpoint (LFP)

[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} = {2,4,5}

[Y02] = [Y01] - - - > {2,4,5} is LFP

[X1] ={2,4,5}

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

We are going to find the LFP

[Y10] = {}

[Y11] = [(a and [next]X1) or [next]Y10] = ([a] intersect PreA(next,[X1])) union PreA(next,Y10) = ({2,4,5} intersect {4}) union {} = {4}

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

[Y12] = [Y11] - - > {4} is LFP

[X2] = {4}

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

We are going to find LFP

[Y20] = {}

[Y21] = [(a and [next]X2) or [next]Y20] = ([a] intersect PreA(next,[X2])) union PreA(next,Y20) = ({2,4,5} intersect {4}) union {} = {4}

[Y22] = [(a and [next]X2) or [next]Y21] = ([a] intersect PreA(next,[X2])) union PreA(next,Y21) = ({2,4,5} intersect {4}) union {4} = {4}

[Y22] = [Y21] - - {4} is LFP

[X3] = {4}

[X3] = [X2] - - > {4} if GFP

It is 1 in [phi] = {4} ? NO, so phi is false this transition system

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

EF AG a and EF AG b

alpha = AG b = vX b and [next] X

beta = EF alpha

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

delta = EF gamma

sigma = delta and beta

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

We are going to compute the greatest fixpoint (GFP)

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

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

[X2] = [b and [next] X0] = [b] intersect PreA(next,[X0]) = {3,4,5} intersect {3,4,5} = {3,4,5}

[X2] = [X1] - - -> {3,4,5} is GFP

[alpha] = {3,4,5}

[beta] = [EF alpha] = [mu X alpha or <next> X]

We are going to compute the least fixpoint

[X0] = {}

[X1] = [alpha or <next>X0] = [alpha] union PreE(next,[X0]) = {3,4,5} union {} = {3,4,5}

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

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

[X3] = [X2] - - >{1,2,3,4,5} is LFP

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

[gamma] = [vX a and [next] X]

We are going to compute GFP

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

[X1] = [a and [next] X0] = [a] intersect PreA(next,[X0]) = {2,4,5} intersect {1,2,3,4,5} = {2,4,5}

[X2] = [a and [next] X1] = [a] intersect PreA(next,[X1]) = {2,4,5} intersect {4} ={4}

[X3] = [a and [next] X1] = [a] intersect PreA(next,[X2]) = {2,4,5} intersect {4} ={4}

[X3] = [X2] - - -> {4} is GFP

[gamma] = {4}

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

We are going to compute LFP

[X0] = {}

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

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

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

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

[X4] = [X3] - - -> {1,2,4} is LFP

[delta] = {1,2,4}

[sigma] = [delta and beta] =[delta] intersect [beta] = {1,2,4} intersect {1,2,3,4,5} = {1,2,4}

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

EX4

while (x<10) do x := x + 5

execution and final state, starting from an initial state where x= 1, S0: x=1

1. Evaluation semantic

Evaluation semantic : Given a program d and a memory state s compute the memory state s’ obtained by execution d in s. (d,s)->s’

(while(x<10) do x:=x+5,S0)->sf

---------------------------------------

(x:=x+5,S0) -> s1 and (while(x<10) do x:=x+5,s1)->sf

S1: x:=6

(while(x<10) do x:=x+5,S0)->sf

---------------------------------------

(x:=x+5,S0) -> S1 and (while(x<10) do x:=x+5,S1)->sf

(while(x<10) do x:=x+5,S1)->sf

---------------------------------------

(x:=x+5,S0) -> S1 and (x:=x+5,S1) -> s2 and (while(x<10) do x:=x+5,s2)->sf

S2: x:=11

(while(x<10) do x:=x+5,S1)->sf

---------------------------------------

(x:=x+5,S0) -> S1 and (x:=x+5,S1) -> S2 and (while(x<10) do x:=x+5,S2)->sf

The while condition is not true anymore so we consider that S2 = sf

1. Transition semantics

Transition semantics: Given a program d and a memory state s compute the memory state s’ and the program d’ that remains to be executed obtained by executing a single step of d in s.

(while (x<10) do x := x + 5, S0) -> (d’,s’)

-------------------------------------------------

(x:=x+5,S0) -> (epsilon,s1)

S1: x=6

(while (x<10) do x := x + 5, S0) -> (d’,s’)

-------------------------------------------------

(x:=x+5,S0) -> (epsilon,S1)

Where (d’,s’)=(epsilon, while (x<10) do x := x + 5, S1)

(epsilon, while (x<10) do x := x + 5, S1) -> (d’’,s’’)

-------------------------------------------------

(epsilon,S1) is final

(while (x<10) do x := x + 5, S1) -> (d’’,s’’)

---------------------------------------------------

(x:=x+5) ->(epsilon,s2)

S2: x=11

(while (x<10) do x := x + 5, S1) -> (d’’,s’’)

---------------------------------------------------

(x:=x+5) ->(epsilon,S2)

Where (d’’,s’’) = (epsilon, while (x<10) do x := x + 5, S2)

(epsilon, while (x<10) do x := x + 5, S2) -> (d’’’,s’’’)

The first condition term is final and also the second because the while condition is not true anymore, we want to check if everything is really final

(epsilon, while (x<10) do x := x + 5, S2)

--------------------------------------------------

(epsilon,S2) is final and (while (x<10) do x := x + 5, S2) is final

EX5

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

q2(x) :- r(x,y), r(x,z), r(x,v), r(y,w), r(w,x), r(z,w), r(v,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

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

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

1. Build canonical database of q1

Now we want to build the canonical database corresponding to q1. Remember that we can solve q1 subseteq q2 if I can extract the database of q1 and check on it the query q2.

q1(c) subseteq q2(c) iff Iq1(c)|=q2(c)

q1(c) and q2(c) are Boolean because we substitute x with constant c

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

delta^I1={c,y,z}

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

c^I1 = c

1. Check that q1(c) subseteq q2(c) iff Iq1|=Iq2. Check if q2 is true over database of q1, so verify if there is an assignment for all free variables

alpha(y) = y because we have c, so (c,y) is in Iq1 and y is good choice

alpha(z) = y same as previous one

alpha(v) = y same as previous one

alpha(w) = z because we have (w,c) so the good choice is to consider z

This is a satisfying assignment alpha

Now we need to check homomorphism. Homomorphism is a mapping between two interpretations, 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 the predicate

Find an homomorphism means that we want to guess a mapping and show that it respect the two properties described 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={c,y,z,v,w}

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

c^I2 = c

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

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

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

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

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

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

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

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