17-06-16

EX1

Alphabet:

P(x), Ar(x), M(x), An(x), Aq(x), C(x), E(x), WE(x), lives(x,y), inhabits(xy), hosts(x,y)

Axioms:

Forall x,y,z. hosts(x,y,z) implies P(x) and Ar(y) and An(z) \\ typing

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

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

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

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

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

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

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

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

EX2

1. The instantiation is not complete because we have some ISA in this UML diagram so we need to apply the following procedure to compute 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) in T

I = Inew

Return I

I = (Obj^I, P^I, Ar^I, An^I, M^I, Aq^I, C^I, E^I, WE^I, lives^I, inhabit^I, hosts^I)

I0:

P^I={londonZoo}

Ar^={sharkPool, fishPool, SavanaArea}

An^I={}

M^I= {}

Aq^I={sawshark,crocodile}

C^I={dolphin, bluewhale}

E^I={}

WE^I={ocean, lagoon}

lives^I={(dolphin,ocean), (bluewhale, ocean), (sawshark, ocean), (crocodile,lagoon)} inhabit^I={(crocodile, ocean), (dolphin,ocean), (bluewhale, ocean), (sawshark, ocean), (crocodile,lagoon)}

hosts^I = {(londonZoo, sharkPool, sawshark),(londonZoo,fishPool,sawshark)}

I1:

P^I={londonZoo}

Ar^={sharkPool, fishPool, SavanaArea}

An^I={}

M^I= { dolphin, bluewhale }

Aq^I={sawshark,crocodile, dolphin, bluewhale }

C^I={dolphin, bluewhale}

E^I={ ocean, lagoon }

WE^I={ocean, lagoon}

lives^I={(dolphin,ocean), (bluewhale, ocean), (sawshark, ocean), (crocodile,lagoon)} inhabit^I={(crocodile, ocean), (dolphin,ocean), (bluewhale, ocean), (sawshark, ocean), (crocodile,lagoon)}

hosts^I = {(londonZoo, sharkPool, sawshark),(londonZoo,fishPool,sawshark)}

I2:

P^I={ londonZoo }

Ar^={ sharkPool, fishPool, SavanaArea}

An^I={ sawshark,crocodile, dolphin, bluewhale, dolphin, bluewhale }

M^I= { dolphin, bluewhale }

Aq^I={sawshark, crocodile, dolphin, bluewhale }

C^I={dolphin, bluewhale}

E^I={ ocean, lagoon }

WE^I={ocean, lagoon}

lives^I={(dolphin,ocean), (bluewhale, ocean), (sawshark, ocean), (crocodile,lagoon)} inhabit^I={(crocodile, ocean), (dolphin,ocean), (bluewhale, ocean), (sawshark, ocean), (crocodile,lagoon)}

hosts^I = {(londonZoo, sharkPool, sawshark),(londonZoo,fishPool,sawshark)}

I3 = I2

So the instantiation is complete

Now we need to check if I3 is correct, this means that I3|= T, so check if all axioms in T are evaluate to true in the interpretation I3.

In this case is correct because all axioms are true. For example, each instance of Aq has only one instance of WE in the association lives. Each instance of An inhabits in at least one instance of E.

1. q(x)<- An(x) and Exists y,y’. inhabits(x,y) and E(y) and inhabits(x,y’) and E(y’) and y noteq y’

q(x):{crocodile}

1. q(x)<- P(x) and forall y. (Exists z. host(x,z,y) implies Aq(y))

q(x):{londonZoo}

1. q()<- Exits x. P(x) and Forall y. (Exists. z host(x,z,y) implies C(y))

q() : false

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.

vX mu Y (a and [next] X) or (b and [next] Y)

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

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

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

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

[Y00] = {}

[Y01] = [(a and [next] X0) or (b and [next] Y00)] = ([a] intersect PreA(next,[X0]) union ([b] and PreA(next,[Y00])) = ({2,4} intersect {1,2,3,4,5}) union {} = {2,4}

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

[Y03] = [(a and [next] X0) or (b and [next] Y02)] = ([a] intersect PreA(next,[X0]) union ([b] and PreA(next,[Y02])) = ({2,4} intersect {1,2,3,4,5}) union {1,2,3,4,5} = {1,2,3,4,5}

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

[X1] = {1,2,3,4,5}

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

It is 1 in phi ={1,2,3,4,5} ? YES, 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.

AG(a implies EX EG b)

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

beta = EX alpha

gamma = a implies beta

delta = AG gamma

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

We are are going to find the greatest fixpoint (GFP) because of the 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> X1] = [b] intersect PreE(next,[X1]) = {3,4,5} intersect {2,4,5} = {4,5}

[X3] = [b and <next> X1] = [b] intersect PreE(next,[X2]) = {3,4,5} intersect {2,4,5} = {4,5}

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

[alpha] = {4,5}

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

[gamma] = [a implies beta] = [not a or beta] = [not a] union [beta] ={1,3,5} union {2,3,4,5} = {1,2,3,4,5}

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

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

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

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

[X1] = [X0] - - > found GFP

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

It is 1 in [delta] ? 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 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,s1),(t1,s2),(t1,s3),(t2,s1),(t2,s2),(t2,s3)}

1. Now we remove all pairs that one that is final and the other is not final

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

1. Now we repeat iteratively that we remove all pairs that one can do one action and the other one cannot copy.

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

* t2 can do b and go to t1 and s3 cannot do b, so I remove (t2,s3)

R3 = {(t2,s2)}

* t1 can do a and go to t2 and s1 can do a and go to s3 but (t2,s3) is not in R2, so I remove (t1,s1).

R4 = {}

* t2 can do b and go to t1 and s2 can do b and go to s1 but (t1,s1) is not in R3 so I remove it.

R5 = R4

S and T are not bisimular because (t1,s1) is not in R5

EX5

q(x) <- Employee(x), Manages(x,y)

Cannot be Green because it is not an Employee and also we canno return null values so x cannot be null1

q(x):{Smith,Brown}