16-12-16

EX1

Alphabet:

T(x), Al(x), SE(x), Ar(x), bonus(x,y), iscontained(x,y), recordedBy(x,y)

Axioms:

Forall x. SE(x) implies Al(x) \\ ISA

Forall x,y. bonus(x,y) implies T(x) and SE(y) \\ typing

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

Forall x,y. bonus(x,y) implies iscontained(x,y) \\ subset

Forall x,y. iscontained(x,y) implies T(x) and Al(y) \\ typing

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

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

Forall x,y. recordedBy(x,y) implies Al(x) and Ar(y) \\ typing

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

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

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 = I

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, T^I, Al^I, SE^I, Ar^I, bonus^I, iscontained^I, recordedBy^I)

I0:

T^I = {t1,t2,t3,t4,t5,t6}

Al^I = {a1,a2,a3}

SE^I = {s1,s2}

Ar^I = {bt,rs}

bonus^I = {(t5,s1),(t6,s2)}

iscontained^I = { (t1,a1), (t2,a1), (t3,a1), (t1,a2), (t4,a2), (t5,a2), (t5,a3)}

recordedBy^I = {(a1,bt),(a2,bt), (a3,rs), (s1,rs), (s2,bt)}

I1:

T^I = {t1,t2,t3,t4,t5,t6}

Al^I = {a1,a2,a3, s1,s2}

SE^I = {s1,s2}

Ar^I = {bt,rs}

bonus^I = {(t5,s1),(t6,s2)}

iscontained^I = { (t1,a1), (t2,a1), (t3,a1), (t1,a2), (t4,a2), (t5,a2), (t5,a3), (t5,s1),(t6,s2)}

recordedBy^I = {(a1,bt),(a2,bt), (a3,rs), (s1,rs), (s2,bt)}

I2=I1

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)<-Ar(x) and Exists y. recordedBy(y,x) and Al(y) and Exists z. SE(z) and recordedBy(z,x) and Exists k. iscontained(k,y) and iscontained(k,z)

q(x): {rs}

1. q(x)<- Ar(x) and forall y. recordedBy(y,x) implies SE(y)

q(x):{}

1. q() <- Exists x. T(x) and forall y. not SE(y) implies iscontained(x,y)

q(): yes

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.muY.((a and <next> X) or ([next]Y))

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

[Y00] = {}

[Y01] = [a and <next> X0) or [next] Y00] = ([a] intersect PreE(next,[X0])) union PreA(next,[Y00]) = ({0,3,4} intersect {0,1,2,3,4}) union {} = {0,3,4}

[Y02] = [a and <next> X0) or [next] Y01] = ([a] intersect PreE(next,[X0])) union PreA(next,[Y01]) = ({0,3,4} intersect {0,1,2,3,4}) union {3,4} = {0,3,4}

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

[X1] = {0,3,4}

[X2] = [muY (a and <next> X0) or [next] Y]

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

[Y10] = {}

[Y11] = [a and <next> X1) or [next] Y10] = ([a] intersect PreE(next,[X1])) union PreA(next,[Y10]) = ({0,3,4} intersect {0,2,3,4}) union {} = {0,3,4}

[Y12] = [a and <next> X1) or [next] Y11] = ([a] intersect PreE(next,[X1])) union PreA(next,[Y11]) = ({0,3,4} intersect {0,2,3,4}) union {3,4} = {0,3,4}

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

[X2] = {0,3,4}

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

It is 0 in [phi] = {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.

AF(EG(a implies AXEX not a))

alpha = EX not a = <next> not a

beta = AX alpha

gamma = a implies beta

delta = EG gamma

sigma = AF delta

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

[beta] = [AX alpha] = [[next] alpha] = PreA(next,[alpha]) = {1,2,4}

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

[delta] = [EG gamma] = [vX gamma and <next> X]

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

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

[X1] = [gamma and <next> X0] = [gamma ] intersect PreE(next, [X0]) = {1,2,4} intersect {0,1,2,3,4} = {1,2,4}

[X2] = [gamma and <next> X1] = [gamma ] intersect PreE(next, [X1]) = {1,2,4} intersect {0,1,2,3} = {1,2}

[X3] = [gamma and <next> X2] = [gamma ] intersect PreE(next, [X2]) = {1,2,4} intersect {0,1,2} = {1,2}

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

[delta] = {1,2}

[sigma] =[AF delta] = [muX delta or [next] X]

We are going to find the least fixpoint LFP because of presence of muX

[X0] = {}

[X1] = [delta or [next] X0] = [delta] union PreA(next,[X0]) = {1,2} union {} = {1,2}

[X2] = [delta or [next] X1] = [delta] union PreA(next,[X1]) = {1,2} union {1,2} = {1,2}

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

[sigma] = {1,2}

It is 0 in [sigma]? NO, so CTL formula is false in this transition system

EX4

Two states are bisimular if they have the same behaviour. This means that:

* Locally look undistinguishable
* Each action done in one state can be also to in the other state

A binary relation R is a bisimulation iff (s,t) in R such that:

* s is final iff t is final
* Forall action a:
  + If s action a s’ then exists t’. t action a t’ and (s’,t’) in R
  + If t action a t’ then exists s’. s action a s’ and (s’,t’) in R

A state s0 of transition system S is bisumular to state t0 of transition system T iff there exists a bisimulation between s0 and t0

The algorithm to compute a bisimulation is the following:

TSS = <A,S,S^0,deltaS, FS>

TTS = <A,T,S^0,deltaT,FT>

1. R = SxT
2. R’= R – {(s,t) such that not (s is in FS eq t is in FT)}
3. While (R noteq R’) do

R = R’

R’’ = R’ – {(s,t) such that Exists s’,a. s action a s’ and not exists t’. t action t’ and (s’,t’) in R’}{(s,t) such that Exists t’,a. t action a t’ and not exists s’. s action s’ and (s’,t’) in R’}

We are going to find if S and T are bisimular

1. Assuming that all state of S are equal to all state of T

R0 = {(t1,q1),(t1,q2),(t1,q3), (t2,q1),(t2,q2),(t2,q3)}

1. Removing that pair that one state is final and the other is not final

R1= {(t1,q1),(t2,q2),(t2,q3)}

1. Repeat iteratively that we remove pairs in which one state can do an action and the other cannot copy.

R2= {(t1,q1),(t2,q2)}

* t2 can do action b and go to t1 and q3 can do action b and go to q2 but (t1,q2) is not in R1 so I remove (t2,q3)

R3 = {(t2,q2)}

* t1 can do action a and go to t2 and q1 can do action a and go to q3 but (t2,q3) is not in R2, so I remove (t1,q1)

R4 = {}

* t2 can do action b and go to t1 and q2 can do action b and go to q1 but (t1,q1) is not in R3, so I remove (t2,q2)

R5 = {}

R5 = R4

S and T are not bisimular because (t1,q1) is not in R5

EX5

q()←contains(x,y),genre(y,z)

q(): yes

q(x,z)←contains(x,y),genre(y,z)

q(x,z): (wywh,progressive)