11-06-15

EX1

Alphabet:

E(x), W(x), D(x), TD(x), Dep(x), R(x), P(x), worksIn(x,y), participates(x,y,z), directs(x,y)

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

Forall x,y,z. participates(x,y,z) implies Dep(x) and P(y) and R(z) \\ typing

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 1<=#{x|directs(x,y)} \\ multi

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

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

EX2

1. To check if instantiation is complete of UML diagram T we need to follow this procedure

Iold=0, Inew = I

While (Inew and Iold are different) do

Iold=Inew

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

Foreach (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) in T

I= Inew

Return I

I = (Obj^I, E^I,W^I, D^I,TD^I,Dep^I,P^I,R^I,worksIn^I,direct^I, partecipates^I)

I0:

E^I ={}

W^I = {John, Mary, Joe}

D^I = {Jim,Rick}

TD^I = {Ann}

Dep^I = {ICT,HR,Management}

P^I={EU123,IT456}

R^I = { projectManager,principalInvestigator,partner}

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

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

participates^I = {(EU123,ICT,projectManager),(EU123,ICT,principalInvestigator),(IT456,ICT,partner),(IT456,Management,projectManager)}

I1:

E^I = {}

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

D^I = {Jim,Rick,Ann}

TD^I = {Ann}

Dep^I = {ICT,HR,Management}

P^I={EU123,IT456}

R^I = { projectManager,principalInvestigator,partner}

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

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

participates^I = {(EU123,ICT,projectManager),(EU123,ICT,principalInvestigator),(IT456,ICT,partner),(IT456,Management,projectManager)}

we add instance “Ann” of TD in W and D because TD has an ISA with W and D.

I2:

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

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

D^I = {Jim,Rick,Ann}

TD^I = {Ann}

Dep^I = {ICT,HR,Management}

P^I={EU123,IT456}

R^I = { projectManager,principalInvestigator,partner}

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

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

participates^I = {(EU123,ICT,projectManager),(EU123,ICT,principalInvestigator),(IT456,ICT,partner),(IT456,Management,projectManager)}

I3 = I2 so the instantiation is complete

Now we need to check if I3 is correct, so what we want to do is to verify if I3|=T , so checking if all axioms in T evaluate to true the interpretation I3

each instance of E is worksIn one and only one Dep, so it is ok. each element of D has directs of at most one Dep, and each Dep is associated only one D, ok. The rest is ok, so the instantiation is correct.

1. q(x)<- P(x) and Exists y. Dep(y) and Exists z. participates(x,y,z) and Exists z’. participates(x,y,z’)

q(x): {EU123,IT456}

1. q(x)<- P(x) and forall Dep(y) and Exists z. participates(x,y,z) and (forall z’. Dep(y,z’) implies z=z’)

q(x): {IT456}

EX3

Model checking a closed mu calculus formula phi over a transition system T = <S,Ra,Pi> (S is set of states, Ra set of transition system and Pi mapping function from a set of proposition 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 it consist in labelling states of T with predicates that are true in them. The extension of least fixpoint and greatest fixpoint are compute using Tarski-Knaster approximates theorem.

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

We have vX so we want to find the greatest fixpoint (GFP)

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

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

We have muY so we are going to compute the least fixpoint (LFP)

[Y00] = {}

[Y01] = [(a and <next>X0) or (not b and <next> Y00)] = ([a] intersect PreE(next,[X0])) union ([not b] intersect PreE(next,[Y00]) = ({4} intersect {1,2,3,4,5}) union ({1,2,3} intersect {}) = {4} union {} = {4}

[Y02] = [(a and <next>X0) or (not b and <next> Y01)] = ([a] intersect PreE(next,[X0])) union ([not b] intersect PreE(next,[Y01]) = ({4} intersect {1,2,3,4,5}) union ({1,2,3} intersect {2,4}) = {4} union {2} = {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]) = ({4} intersect {1,2,3,4,5}) union ({1,2,3} intersect {1,2,4}) = {4} union {1,2} = {1,2,4}

[Y04] = [(a and <next>X0) or (not b and <next> Y03)] = ([a] intersect PreE(next,[X0])) union ([not b] intersect PreE(next,[Y03]) = ({4} intersect {1,2,3,4,5}) union ({1,2,3} intersect {1,2,4,5}) = {4} union {1,2} = {1,2,4}

[Y04] = [Y03] - - > {1,2,4} is LFP

[X1] = {1,2,4}

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

We have muY so we are going to compute the least fixpoint (LFP)

[Y10] = {}

[Y11] = [(a and <next>X1) or (not b and <next> Y10)] = ([a] intersect PreE(next,[X1])) union ([not b] intersect PreE(next,[Y10]) = ({4} intersect {1,2,4,5}) union ({1,2,3} intersect {}) = {4} union {} = {4}

[Y12] = [(a and <next>X1) or (not b and <next> Y11)] = ([a] intersect PreE(next,[X1])) union ([not b] intersect PreE(next,[Y11]) = ({4} intersect {1,2,4,5}) union ({1,2,3} intersect {2,4}) = {4} union {2} = {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]) = ({4} intersect {1,2,4,5}) union ({1,2,3} intersect {1,2,4}) = {4} union {1,2} = {1,2,4}

[Y14] = [(a and <next>X1) or (not b and <next> Y13)] = ([a] intersect PreE(next,[X1])) union ([not b] intersect PreE(next,[Y13]) = ({4} intersect {1,2,4,5}) union ({1,2,3} intersect {1,2,4,5}) = {4} union {1,2} = {1,2,4}

[Y14] = [Y13] - - > {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 formula is true in 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 is set of initial states, R set of transitions, AP set of atomic propositions, L is labelling function L:S->2^AP) and CTL formula phi we want to verify if KM,s|= phi where s is state of S. With model checking we return a subset of S in which each state satisfied phi. To compute it we need to exploit the syntactic structure of CTL formula, in particular, we translate each CTL sub formula into mu calculus formula and then apply the labelling algorithm to find their extensions.

EG AF a or EF AG b

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

beta = EF alpha

gamma = AF a = muX a or [next] X

delta = EG gamma

sigma = delta or beta

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

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

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

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

[X2] =[b and [next] X1] = [b] intersect PreA(next,[X1]) ={4,5} intersect {2,3} = {}

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

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

[alpha] = {}

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

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

[X0] = {}

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

[X1] = [X0] - - > {} is LFP

[beta] = {}

[gamma] = [muX a or [next] X]

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

[X0] = {}

[X1] = [a or [next]X0] = [a] union PreA(next,[X0]) = {4} union {} = {4}

[X2] = [a or [next]X1] = [a] union PreA(next,[X1]) = {4} union {2} = {2,4}

[X3] = [a or [next]X2] = [a] union PreA(next,[X2]) = {4} union {2} = {2,4}

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

[gamma] = {2,4}

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

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

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

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

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

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

[delta] = {2,4}

[sigma] = [delta or beta] = [delta] union [beta] = {2,4} union {} = {2,4}

It is 1 in [sigma] ? NO, so CTL formula is false in this transition system

EX4

wp(d,Q) = {s| forall s’ ((d,s)->s’)->s’|=Q}

All states s such that the execution of the program d in the state s gives s’ that satisfied the post-condition Q. wp gives us the minimum condition such that we will achieve Q by executing the program d. Since we don’t have a while instruction we can compute the weakest precondition automatically, starting from below and going backward.

{x>=50 and y+2+y = 100} = {x>=50 and 2y= 98} = {x>=50 and y = 49} is the wp

x := y + 2;

[{y<0 and x=50 and x<50} or {y>=0 and x=100 and x<50} ] or {x>=50 and x+y = 100} = [{false} or {false}] or {x>=50 and x+y = 100} = {x>=50 and x+y = 100}

if (x < 50) then {

{y<0 and x=50} or {y>=0 and x=100}

if (y < 0) then

{2\*x=100} = {x=50}

x := 2\*x;

{x=100}

{x=100}

else y := y\*y

{x=100}

}

{x+y = 100}

else x := x + y;

{x=100}

{x=100}

y := y\*y

{x=100}

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,y).

We need to verify if q1 is contained in q2, this means that forall x. q1(x) implies q2(x) is valid. Valid means that forall I,alpha |= forall x. q1(x) implies q2(x) with I interpretation and alpha assignment.

In FOL validity is undecidable but we can use conjunctive queries to make it satisfiable because we can transform queries in databases.

We need 3 steps:

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

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