22-01-15

EX1

Aphabet:

G(x), M(x), includes(x,y), appears(x,y), C(x),SM(x), dedicatedTo(x,y), points(x,y,z), Int(x)

Axioms:

Forall x,y. includes(x,y) implies G(x) and M(y) \\ typing

Forall x,y,z. points(x,y,z) implies includes(x,y) and Int(z) \\ typing

Forall x. G(x) implies 3<=#{y|includes(x,y)} \\ multi

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

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

Forall x,y. dedicatedTo(x,y) implies SM(x) and C(y) \\ typing

Forall x,y. appears(x,y) implies M(x) and C(y) \\ typing

Forall x,y. dedicatedTo(x,y) implies appears(x,y) \\ subset

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

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

EX2

1. To complete instantiations of UML diagram T we use this procedure:

I\_old = 0, I\_new = I

While(I\_new is different from I\_old) do

I\_old=I\_new

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

For each (a in A^{I\_new}) do

B^{I\_new} = B^{I\_new} union {a}

Similar for each subset constraints forall x,y. P(x,y)->R(x,y) in T

I= I\_new

Return I

I = {Obj^I,M^I, SM^I,C^I,appears^I, dedicatedTo^I}

I0:

M^I={artica,bush}

SM^I={city,desert}

C^I={adrian,bob,charline}

Appears^I = {(adrian,artica),(adrian,bush),(adrian,desert),(bob,city),(charline,artica)}

dedicatedTo^I={(adrian,city),(charline,desert)}

I1:

M^I={artica,bush,city,desert}

SM^I={city,desert}

C^I={adrian,bob,charline}

Appears^I = {(adrian,artica),(adrian,bush),(adrian,desert),(bob,city),(charline,artica),(adrian,city),(charline,desert }

dedicatedTo^I={(adrian,city),(charline,desert)}

I2 = I1

The instantiation is complete

To check if the instantiations I 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.

In this case the instantiation I2 is correct because each map has at least one character in the appear associations, then element of specialmap has one and only one character in the association dedicatedTo. Each elements in dedicatedTo is contained in appears.

a) C(x) and Exists y. appears (x,y) and Map(y)

b) C(x) and Forall y. (Map(y) implies appears(x,y)

EX3

Model checking a closed mu calculus formula phi over transition system T = <S,Ra,Pi> (S is set of states, Ra set of transitions and Pi mapping function from a set of propositions P to a subset of S) means that we want to verify that the initial state of T is in the extension of phi over T. Model checking return a subset of S in which each state satisfied phi. To compute this subset we need to use the labelling algorithm, that consist in labelling the states of T with predicates that are true in them. The extensions of least fixpoint and greatest fixpoint are computed by applying the Tarksi-Knaster approximates theorem.

vXmuY((b and <next>X) or<next>Y)

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

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

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

We have muY so now we want to compute the least fixpoint LFP

[Y00] = {}

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

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

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

[Y03] =[Y02] - - > {1,2,3,4} is LFP

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

[X2] = [muY((b and <next>X1) or<next>Y)]

We want to find the least fixpoint

[Y10] = {}

[Y11] = [((b and <next>X1) or<next>Y10)] = ([b] intersect PreE(next,[X1])) union PreE(next,[Y10]) = ({3,4} intersect {1,2,3,4}) union {} = {3,4}

[Y12] = [((b and <next>X1) or<next>Y11)] = ([b] intersect PreE(next,[X1])) union PreE(next,[Y11]) = ({3,4} intersect {1,2,3,4}) union {1,2,3,4} = {1,2,3,4}

[Y13] = [((b and <next>X1) or<next>Y12)] = ([b] intersect PreE(next,[X1])) union PreE(next,[Y12]) = ({3,4} intersect {1,2,3,4}) union {1,2,3,4} = {1,2,3,4}

[Y13] = [Y12] - - > {1,2,3,4} is LFP

[X2] = {1,2,3,4}

[X2] = [X1] - - -> {1,2,3,4} is GFP

It is 1 in {1,2,3,4} ? YES, so the phi mu calculus formula is true in the transition system

Now we want to do model check with CTL formula. Given a Kripke model KM = <S,I,R,AP,L> (S set of states, I set of initial states, R is set of transition systems, Ap set of atomic propositions, L labelling function L:S->2^AP) and CTL formula phi we want to verify that KM,s |= phi, where s is state of S. With model checking we return a subset of S in which each state satisfies phi. To compute it we need to exploit the syntactic structure of CTL formula, in particular, we translate CTL sub formula into mu calculus formula and then we apply labelling algorithm to them to find their extensions.

EG(AX(not a or AF b))

alpha = AF b = muX b or [next] X

beta = not a or alpha

gamma = AX beta

delta = EG gamma

[alpha] = [muX b or [next] X]

We are going to compute the least fixpoint

[X0] ={}

[X1] = [b or [next] X0] = [b] union PreA(next,[X0]) = {3,4} union {} = {3,4}

[X2] = [b or [next] X1] = [b] union PreA(next,[X1]) = {3,4} union {3,4} = {3,4}

[X2] = [X1] - - > {3,4} LFP

[alpha] = {3,4}

[beta] = [not a or alpha] = [not a] union [alpha] = {1,3} union {3,4} = {1,3,4}

[gamma] =[AX beta] = [[next] beta] = PreA(next,[beta]) = {3,4}

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

We are going to find GFP

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

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

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

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

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

EX4

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

All states s such that the execution of program d in the state s gives s’ that satisfies the post-condition Q. The wp give us the minimum condition such that you will achieve Q by executing d. Since we don’t have “while” instruction in this program we can compute wp automatically

{x>50 and y>0 and 50+y=y} or {x=0} = {false} or {x=0} = {x=0} is wp

x := 50 + y;

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

if (x > 50) then {

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

if (y > 0) then

{x=y}

x := x - y;

{x=0}

{x=0}

else y := -y

{x=0}

}

{x+y = 0}

else x := x + y;

{x=0}

{x=0}

y := y + 50

{x=0}

EX5

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

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

First we want to check if q1is contained in q2 so, this means that we want to check if the q(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 3 steps:

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

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

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

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

q1(c) subseq q(2) iff Iq1(c)|=q2(c)

q1(c) and q2(c) are boolean now because I substitute x with constant c

1. Build canonical database of q1

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

delta^I = {c,y,z}

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

c^I = c

Now we need to check that q2 is true over this database, this means that I need to find an assignment forall free variables y,z,v,w that makes q2 true

alpha(y) = y

alpha(z) = y

alpha(v) = y

This is a satisfying assignment, i.e. makes formula true

Now we need to show homomorphism. Homomorphism is a mapping between two interpretations, between of 2 domains h=delta^I-> delta^J such that:

* h(c^I) = c^J
* (x,y) in e^I implies (h(x),h(y)) in e^J

Find a homomorphism is to guess a mapping and show that it respect these two properties above. But remember that there is a theorem that says that if you have an assignment alpha , which is a satisfying assignment, you can transform alpha in 2 homomorphism between two canonical databases Iq1|=Iq2 iff h:Iq2 implies Iq1

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

Delta^I2 = {c,y,z,v}

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

h(c) = h(c^Iq2) = c

h(y) =alpha (y) =y

h(z) =alpha (z) =y

h(v) =alpha (v) =y

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

(c,y) in e^I2 -- > (h(c),h(y)) in e^I ok because (c,y) is in Iq1

(y,z) in e^I2 -- > (h(y),h(z)) in e^I ok because (y,y) is in Iq1

(c,z) in e^I2 -- > (h(c),h(z)) in e^I ok because (c,y) is in Iq1

(c,v) in e^I2 -- > (h(c),h(v)) in e^I ok because (c,y) is in Iq1

(v,z) in e^I2 -- > (h(v),h(z)) in e^I ok because (y,y) is in Iq1

(v,y) in e^I2 -- > (h(v),h(y)) in e^I ok because (y,y) is in Iq1