25-01-17

EX1

Alphabet:

GL(x), M(x), C(x), SM(x), appears(x,y), mainCharacter(x,y), includes(x,y), points(x,y,z), Int(x)

Axioms:

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

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

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

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

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

Forall x,y. mainCharacter(x,y) implies SP(x) and C(y) \\ typing

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

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

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

Forall x. M(x) implies 1<=#{y|appears(x,y)}<=9 \\ multi after refinements

EX2

The instantiation is not complete because we have ISA, so we need to following this procedure to complete the instantiation given a UML diagram T.

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^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,GL^I,includes^I, M^I,SM^I,C^I, appears^I,mainCharacter^I)

I0:

GL^I = {}

includes^I= {}

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)}

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

I1:

GL^I = {}

includes^I= {}

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)}

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

I2=I1 so the instatiation is complete

Now to check is I2 is correct we need to verify that I2|=T, so check if all axioms of T evaluate true in interpretation I2.

Each instance of SM is also instance of M. Each instance of M has at most 9 instance of C that appear, it is ok because artica has 2, bush 1, city 2 and desert 2. Each instance of SM has one and only one mainCharacter, so only one instance of C. Infact, city has adrian and desert has charline. Each instance of mainCharacter is instance of appears.

The instantiation is correct.

1. q(x)<- M(x) and Exists y,y’,y’. C(y) and appears(y,x) and C(y’) and appears(y’,x) and C(y’’) and appears(y’’,x) and y noteq y’ and y’ noteq y’’ and y noteq y’’

q(x):{}

1. q(x)<- C(x) and Forall y. y appear(x,y) implies mainCharacter(x,y)

q(x):{}

1. q()<- Exists x. M(x) and forall y. appear(y,x) implies C(y)

q(): false

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.

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

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

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

[X1] = [mu Y (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]) = ({2,4,5} intersect {1,2,3,4,5}) union {} = {2,4,5}

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

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

[Y04] = [(a and <next> X0) or [next] Y03] = ([a] intersect PreE(next,[X0])) union PreA(next,[Y03]) = ({2,4,5} intersect {1,2,3,4,5}) union {1,2,3,4,5} = {1,2,3,4,5}

[Y04] = [Y05] - - >found LFP

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

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

It is 1 in [phi ]= {1,2,3,4,5} ? YES, so phi is satisfied by 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.

EF(not a implies EX AG b)

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

beta = EX alpha

gamma = not a implies beta

delta = EF gamma

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

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

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

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

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

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

[alpha] = {3,4}

[beta] = [EX alpha] = [<next> alpha] = PreE(next,[alpha]) = {1,2,3,4}

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

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

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

[X0] ={}

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

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

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

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

It is 1 in [delta]? YES, so the CTL formula is true in this transition system

EX4

Hoar triple {P} while g do S {Q} is correct cannot be computed automatically but we can use inference rule.

We assume to have a candidate invariant I and we need to show:

1. P implies I
2. {I and g} S {I} = {I and g} implies wp(S,I)
3. {I and not g} implies Q

If candidate satisfied this 3 properties it is called invariant.

If we have the candidate these operation are automatic, otherwise it difficult to manage with it. The problem of checking if the hoare triple is true or not is undecidable, so there is not sound or complete technique to do this or to generate all possible invariants. We can only use sound technique to verify whether candidate is invariant or not.

Candidate : (i+j)=9

{i=0 AND j=9} while(i<10) do (i:= i+1; j=j-1) {j<0}

* P implies I

{i = 0 and j=9} implies {i+j=9}

Yes, because if 0+9 is always equal of 9

* {I and not g } implies Q

{i+j=9 and i>= 10} implies {j<0}

Yes because if i is major than 10 to have i+j = 9 we need to have j minor than 0

* Here first we compute wp(S,I)

{i+1+j-1=9} = {i+j=9} is wp

i=i+1

{i+j-1=9}

j=j-1

{i+j=9}

So {I and g} implies wp(S,I)

{i+j=9 and i<10} implies {i+j=9}

Yes, the condition is satisfied also because the thesis is also equal to a part of hypothesis

So i+j=9 is invariant

EX5

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

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

We need to check if q1 is contained in q2, this means that we want to check if q1 (x) implies q2(x) is valid. Validity means that forall I,apha |= forall x. q1(x) implies q2(x)

Where I interpretation and alpha assignment.

Validity is undecidable in FOL but with CQ we can make it decidable because we can transform them in databases

There are 3 steps:

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

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

a2(c) :- edge(c,y), edge(y,z), edge(c,v), edge(v,z), edge(v,y)

1. Build canonical database of q1. Remember that we can solve q1(x) subseteq q2(x) if we can extract of q1 and check on q2

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

delta^I1 = {c,t,z}

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

c^I1 = {c}

1. Check if q2 is true over database of q1, so verify if there is a assignment for all free variables

alpha(y) =y

alpha(z) = z

alpha(v) = y

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={y,z,v,c}

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

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

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

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

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

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