18-02-14

EX1

Alphabet:

A(x), B(x),C(x),D(x), Rac(x,y), Rbd(x,y)

Axioms:

Forall x. B(x) implies A(x) \\ ISA

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

Forall x. D(x) implies B(x) \\ ISA

Forall x. C(x) implies not D(x) \\ disjunction

Forall x. B(x) implies C(x) or D(x) \\ complete

Forall x,y. Rac(x,y) implies A(x) and C(y) \\ typing

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

Forall x,y. Rbd(x,y) implies B(x) and D(y) \\ typing

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

EX2

Model checking a closed mu calculus formula phi over a transition T = <S,Ra,Pi>, (where S is set of states, Ra is set of transitions and Pi is a 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 (in S) is in the extension of phi over T, given a valuation V.

We can compute it with labelling algorithm, that consist in labelling states of T with predicates that are true in them. The extensions of least fixpoint and greatest fixpoint are computed applying the Tarski-Knaster approximate theorem.

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

We have vX and this means that we want to find the greatest fixpoint (GFP)

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

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

muY identify that we are going to find the least fixpoint (LFP)

[Y00] = {}

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

[Y01] = [Y00] -- > {} is LFP

[X1] = {}

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

Also in this case we want to find LFP, so

[Y10] = {}

[Y11] = [(a or <next> X1) and [next] Y10] = ([a] union PreE(next,[X0])) intersect PreA(next,[Y10]) = ({2,4} union {1,2,3,4}) intersect {} = {}

[Y11] = [Y10] -- > {} is LFP

[X2] = {}

[X2] = [X1] -- > {} is GFP

We want to know if 1 is in [phi] = {}, so NO and this means that phi is FALSE in this transition system

Regarding of model checking in CTL, we need to have a Kripke model (KM) = <S,I,R,AP,L> (S is set of state, I is set of initial states, R is set of transitions, AP is set of atomic proposition and L is mapping function L:S->2^AP) and a CTL formula phi, so model checking means that KM, s |= phi where s is state of S. Model checking return a set of states, a subset of S. To compute this set we need to exploit syntactic structure of CTL formula, in particular we translate its sub formulas into mu calculus formulas and then applying to each mu calculus sub formula the labelling algorithm to find their extension.

EG(not a implies AXAF a)

alpha = AF a

beta = AX alpha

gamma = not a implies beta

delta = EG gamma

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

Mu identify that we want to compute the least fixpoint (LFP)

[X0] = {}

[X1] = [a or [next] X0] = [a] union PreA(next,[X0]) = {2,4} union { } = {2,4}

[X2] = [a or [next] X1] = [a] union PreA(next,[X1]) = {2,4} union {1,3,4} = {1,2,3,4}

[X3] = [a or [next] X2] = [a] union PreA(next,[X2]) = {2,4} union {1,3,4} = {1,2,3,4}

[X3] = [X2] so {1,2,3,4} is LFP

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

[beta] = [AX alpha] = [[next] alpha] = {1,3,4}

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

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

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

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

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

[X1] = [X0] -- > {1,2,3,4} is GPF

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

Is [delta] true in transition system T? YES because 1 is in the [delta], so CTL is TRUE in this transition system

EX3

1. q() <- Exists x. Employee(x) and MSc(x) and Exists y. Manages(x,y) and MSc(y) \\ IT IS A CQ because it is composed only on existential quantifier
2. q() <- Exists x. Employee(x) and MSc(x) and Exists y,y’. Manages(x,y) and MSc(y) and Manages(x,y’) and y’ noteq y \\ IT is a CQ
3. q() <- Exists x. Employee(x) and MSc(x) and Exists y. Manages(x,y) and MSc(y) and Exists y’. Manages(x,y’) \\ IT is a CQ
4. q() <- Exists x. Employee(x) and (Forall y. Manages(x,y) implies MSc(y)) \\ IT is not a CQ
5. q() < - Exists x. Employee(x) and (Forall y. MSc(y) implies Manages(x,y)) \\ IT is not a CQ

EX4

q(x) <- Employee(x) and Exists y. Manages(x,y)

q(x): {Smith, Brown}

EX5

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

All the states s such that the execution of the program d in the state s, give a state s’ that satisfies the post-conditions Q. So, the wp gives you the minimum condition such that you will achieve Q by executing d. Since we don’t have the while in this program we can compute wp automatically, starting from below and going backward.

*{y>0 and y+1=-y} or {y=-100} = {y>0 and y=-1/2} or {y=-100} = {false} or {y=-100} = {y=-100} is wp*

x:=y+1

*{y>0 and x=-y} or {y<=0 and y=-100} = {y>0 and x=-y} or {y=-100}*

if (y>0) then

*{x+y=0}= {x=-y}*

x:=x+y

*{x:=0}*

*{y+100 = 0}={y=-100}*

else x:=y+100

*{x=0}*

*{x+y=y} = {x=0}*

x:=x+y

*{x=y}*