12-04-14

EX1

Alphabet:

A(x), B(x), C(x), D(x), E(x), Rcd(x,y), Qae(x,y)

Axioms:

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

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

Forall x. B(x) implies not C(x) \\ disjoint

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

Forall x E(x) implies D(x) \\ ISA

Forall x,y. Qae(x,y) implies Rcd(x,y) \\ subset

Forall x,y. Qae(x,y) implies A(x) and E(y) \\ typing

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

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

Forall x,y. Rcd(x,y) implies C(x) and D(y) \\ typing

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

Forall y D(y) implies 1<=#{x|Rcd(x,y)} after refinements \\ multi

EX2

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 and Pi is mapping function from a set of proposition P to a subset of S) means to verify if the initial state s of T is in the extension of phi over T, given a valuation V. When we apply model checking we return a subset of S in which, each state satisfies phi, so to compute this subset we need to apply 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 using the Tarski-Knaster approximates theorem.

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

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

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

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

muY so we need to compute the least fixpoint (LFP)

[Y00] = {}

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

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

[Y03]= [(b and <next> X0) or <next> Y02] = ([b] intersect PreE(next,[X0])) union PreE(next,[Y01]) = ({4} intersect {1,2,3,4}) union {2,3} = {2,3,4}

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

[Y05] =[(b and <next> X0) or <next> Y04] = ([b] intersect PreE(next,[X0])) union PreE(next,[Y04]) = ({4} intersect {1,2,3,4}) union {1,2,3,4} = {1,2,3,4}

[Y05] = [Y04] - - > {1,2,3,4} is LFP

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

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

So the entire formula [phi] = {1,2,3,4}. It is 1 in {1,2,3,4} YES, so the phi is true in 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.

AG (AF a and EF b and EG not b)

alpha = EG not b = vX not b and <next> X

beta = EF b = mu X b or <next> X

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

delta = alpha and beta and gamma

sigma = AG delta

[alpha] = [vX not b and <next> X]

vX means that we want to find the greatest fixpoint (GFP) so

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

[X1] = [not b and <next>X0] = [not b] intersect PreE(next,[X0]) = {1,2,3} intersect {12,3,4} = {1,2,3}

[X2] = [not b and <next>X1] = [not b] intersect PreE(next,[X1]) = {1,2,3} intersect {1,2,3,4} = {1,2,3}

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

[alpha] = {1,2,3}

[beta] = [mu X b or <next> X]

We are going to find the least fixpoint (LFP)

[X0] = {}

[X1] =[b or <next> X0] = [b] union PreE(next,[X0]) = {4} union {} = {4}

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

[X3] =[b or <next> X2] = [b] union PreE(next,[X2]) = {4} union {2,3} = {2,3,4}

[X4] =[b or <next> X3] = [b] union PreE(next,[X3]) = {4} union {1,2,3} = {1,2,3,4}

[X5] =[b or <next> X4] = [b] union PreE(next,[X4]) = {4} union {1,2,3,4} = {1,2,3,4}

[X5] = [X4] - - > {1,2,3,4} is LFP

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

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

We are going to find least fixpoint (LFP)

[X0] ={}

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

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

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

[X3] = [X2] - - > {1,2} is LFP

[gamma] = {1,2}

[delta] = [alpha and beta and gamma] = {1,2,3} intersect {1,2,3,4} intersect {1,2} = {1,2}

[sigma] =[vX delta and [next] X]

We are going to find greatest fixpoint GFP

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

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

[X2] = [delta and [next] X1] = [delta] intersect PreA(next,[X1]) = {1,2} intersect {1,4} = {1}

[X3] = [delta and [next] X2] = [delta] intersect PreA(next,[X2]) = {1,2} intersect {4} = {}

[X4] = [delta and [next] X3] = [delta] intersect PreA(next,[X3]) = {1,2} intersect {} = {}

[X4] = [X3] - - > GFP is {}

It is 1 in {}? NO, so CTL formula is false in this transition system

EX3

1. q() <- Exists x. Supplier(x,NY) and Exists y. Item(y,Blue) and Exists z. Sells(x,y,z) \\ IT is a CQ
2. q()<- Exists x. Supplier(x,NY) and Exists y. Item(y,Blue) and Exists z. Sells(x,y,z) and Exists k. Item(k,Blue) Exists j. Sells (x,k,j) and k noteq y \\ IT is a CQ
3. q()<-Exists x. Supplier(x,NY) and Exists y. Item(y,Blue) and Exists z. Sells(x,y,z) and forall k. (Item(k,Blue) and Exists j. Sells(x,k,j) implies k = y)
4. q()<-Exists x. Supplier(x,NY) and Forall y. Item(y,Blue) and Exists z. Sells(x,y,z)
5. q(x,x’)<-Exists y. Supplier(x,y) and Exists z,k. Item(z,k) and Exists y’. Supplier(x’,y’) and Exists z’,k’. Item(z’,k’) and Exists j. Sells(x,z,j) and Exists j’. Sells(x’,z’,j’) and j minor j’
6. q(x,x’)<-Exists y,k,j. Supplier(x,y) and Exists y’,k’,j’. Supplier(x’,y’) and Forall z,z’ ((Item(z’,j’) and Sells(x’,z’,k’)) implies Sells(x,z’,k) and k minor k’)

EX4

1. q(x)←Sells(x,y),Item(y,z)

null5 is not belong to item so I eliminate it, this means that x could not be White. Elements that satisfy the formula are Smith, Brown, null2, Green and null4 but we cannot have null in the result so

q(x) = {Smith, Brown, Green}

1. q(x,z)←Sells(x,y),Item(y,z)

similar of previous case but in this case the pairs that we have to consider are (Smith,red), (Brown,null11), (Green,null10) but we cannot have null in the results

q(x) = {(Smith, red)}

EX5

Check whether a Hoare triple {P} while g do S {Q} is correct cannot be done automatically, we need to use inference rule

We assume to have a candidate variant 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 I satisfies 3 premises above then it called invariant

If the invariant I is given, these operation are automatic, otherwise it could be difficult to manage with. The problem of checking if a triple with while is true or not is undecidable, there is not sound and complete technique to do this or to generate all possible invariants for the problem. We have only a sound technique to check whether a candidate is an invariant or not.

If it is not the case, we cannot say that the triple is false but at the same time we are not sure if an invariant exists.

Candidate invariant: i<= 64

{i=1} while(i<64) do i:= i\*2 {i=64}

* P implies I

{i=1} implies {i<=64} \\ this is true because i=1 is minor of 64

* {I and not g} implies Q

{i <= 64 and i>=64 } implies {i=64} \\ this is true

* First we compute the wp(S,I)

{i\*2<=64} = {i<=32}is the wp

i=i\*2

{i<=64}

Then we will show {I and g} implies wp(S,I)

{i<=64 and i<64} implies {i<=32}

{i<64} implies {i<=32}

This condition is not satisfied because if i is less than 64 it doesn’t mean that i is always less than 32.

So {i<=64} is not an invariant, we cannot say that the triple is false