18-01-18

EX1

Alphabet:

C(x),BC(x), P(x), S(x), contract(x,y,z), cost(x,y,z,k), provides(x,y), Real(x)

Axioms:

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

Forall x,y,z. contract(x,y,z) implies C(x) and P(y) and S(z) \\ typing

Forall x,y,y’,z. contract(x,y,z) and contract(x,y’,z) implies y = y’ \\ key

Forall x,y,z,k. cost(x,y,z,k) implies contract(x,y,z) and Real(k) \\ typing

Forall x,y,z. contract(x,y,z) implies 1<=#{k|cost(x,y,z,k)}<=1 \\ multiplicity

Forall x,y. provides(x,y) implies P(x) and S(y) \\ typing

Forall x. P(x) implies 1<=#{y|provides(x,y)}<=10 \\ multiplicity

Forall y. S(y) implies 1<=#{x|provides(x,y)} \\ multiplicity

EX2

1. This instantiation is not complete because we have an ISA so we need to apply the following procedure to obtain a complete 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) implies B(x) in T) do

For each (a in A^Inew) do

B^Inew = B^Inew union {a}

Similar for each subset constraint forall x,y. P(x,y) implies R(x,y)

I = Inew

Return I

I = (Obj^I, C^I,BC^I,contract^I,P^I,S^I, cost^I)

I0:

C^I = {c1,c2,c3,c4}

BC^I={b1,b2,b3}

contract^I/cost^I = {(c1,s1,p1,90), (c1,s2,p1,80), (c1,s3,p1,50), (b2,s1,p2,170), (b2,s2,p2,100)}

P^I = {p1,p2}

S^I={s1,s2,s3}

provides^I = {(p1,s1), (p1,s2), (p1,s3), (p2,s2)}

I1:

C^I = {c1,c2,c3,c4, b1,b2,b3}

BC^I={b1,b2,b3}

contract^I/cost^I = {(c1,s1,p1,90), (c1,s2,p1,80), (c1,s3,p1,50), (b2,s1,p2,170), (b2,s2,p2,100)}

P^I = {p1,p2}

S^I={s1,s2,s3}

provides^I = {(p1,s1), (p1,s2), (p1,s3), (p2,s2)}

I2 = I1

So the instantiation is complete

Now we need to check if it is correct, so I2 |= T, this means that we want to verify if all axioms in T are evaluate true in this interpretation I2.

Each instance of BC is also instance of C, so it is correct because of ISA. In the association contract we need to have one instance of C, one of P and another of S and there will be only one tuple contains this 3. For example, for (c1,s1,p1) we cannot have another contract with these 3. So it is satisfied. For each instance of S it will be provided alt least one instance of P, ok. And for each instance of P provides al least 1 and at most 10 instance of S. ok

The instantiation is correct

1. q()<- Exists x. C(x) and Exists y,z’,z. S(y) and P(z) and P(z’) and contract(x,y,z) and contract(c,y,z’) and z’ noteq z

q():false

1. q(x)<- C(x) and (forall y. (exists z. S(y) and P(z) and contract(x,y,z)) and forall y’(exists z’. contract(x,y’,z’) and S(y) and P(z’) )) implies y=y’

q(x): {}

1. q(x) <- C(x) and forall y. (exists z,z’. contract(x,y,z) and S(y) and contract(x,y,z’)) implies z=z’

q(x):{c1,b2}

EX3

Model checking a closed mu calculus formula phi over T=<S,Ra,Pi> (S set of states, Ra set of transitions and Pi a mapping function from a set of proposition P to a subset of S) means that we want to check 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 element satisfies phi. To compute it we apply 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 using Tarski-Knaster approximates theorem.

vXmuY((a and [next] X) or (b and [next] Y)

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

[X0] = {0,1,2,3,4}

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

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

[Y00] = {}

[Y01] = [(a and [next] X0) or (b and [next]Y00)] = ([a] intersect PreA(next,[X0])) union ([b] intersection PreA(next,[Y00])) = ({0,1,2} intersect {0,1,2,3,4}) union ({0,3,4} intersect {}) = {0,1,2}

[Y02] = [(a and [next] X0) or (b and [next]Y01)] = ([a] intersect PreA(next,[X0])) union ([b] intersection PreA(next,[Y01])) = ({0,1,2} intersect {0,1,2,3,4}) union ({0,3,4} intersect {1,4}) = {0,1,2,4}

[Y03] = [(a and [next] X0) or (b and [next]Y02)] = ([a] intersect PreA(next,[X0])) union ([b] intersection PreA(next,[Y02])) = ({0,1,2} intersect {0,1,2,3,4}) union ({0,3,4} intersect {1,2,4}) = {0,1,2,4}

[Y03] = [Y02] - - >found LFP

[X1] = {0,1,2,4}

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

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

[Y10] = {}

[Y11] = [(a and [next] X1) or (b and [next]Y10)] = ([a] intersect PreA(next,[X1])) union ([b] intersection PreA(next,[Y10])) = ({0,1,2} intersect {1,2,4}) union ({0,3,4} intersect {}) = {1,2}

[Y12] = [(a and [next] X1) or (b and [next]Y10)] = ([a] intersect PreA(next,[X1])) union ([b] intersection PreA(next,[Y10])) = ({0,1,2} intersect {1,2,4}) union ({0,3,4} intersect {1,2}) = {1,2}

[Y12] = [Y11] - - > found LFP

[X2] = {1,2}

[X3] = [ muY((a and [next] X1) or (b and [next] Y)]

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

[Y20] = {}

[Y21] = [(a and [next] X2) or (b and [next]Y20)] = ([a] intersect PreA(next,[X2])) union ([b] intersection PreA(next,[Y20])) = ({0,1,2} intersect {1}) union ({0,3,4} intersect {}) = {1}

[Y22] = [(a and [next] X2) or (b and [next]Y21)] = ([a] intersect PreA(next,[X2])) union ([b] intersection PreA(next,[Y21])) = ({0,1,2} intersect {1}) union ({0,3,4} intersect {}) = {1}

[Y22] = [Y21] - - > found LFP

[X3] = {1}

[X4] = [ muY((a and [next] X3) or (b and [next] Y)]

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

[Y30] = {}

[Y31] = [(a and [next] X3) or (b and [next]Y30)] = ([a] intersect PreA(next,[X2])) union ([b] intersection PreA(next,[Y30])) = ({0,1,2} intersect {}) union ({0,3,4} intersect {}) = {}

[Y31] = [Y30] - - > found LFP

[X4] = {}

[X5] = [ muY((a and [next] X4) or (b and [next] Y)]

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

[Y40] = {}

[Y41] = [(a and [next] X4) or (b and [next]Y40)] = ([a] intersect PreA(next,[X2])) union ([b] intersection PreA(next,[Y40])) = ({0,1,2} intersect {}) union ({0,3,4} intersect {}) = {}

[Y41] = [Y40] - - > found LFP

[X5] = {}

[X5] = [X4] - - > found GFP

It is 1 in {}? NO, so the phi formula is not satisfied by this transition system

Now we are going to do model checking with CTL formula. Given a Kripke model (KM) = <S,I,R,AP,L> (S is set of states, I set of initial state, R se of transitions, AP set of atomic proposition and L labelling function) and a CTL formula phi, we want to check if KM,s |= phi where s is state of S. With model checking we return a subset of S in which each element satisfied phi, and to compute it we need to exploit the syntactic structure of CTL formula, in particular, we translate each sub formula of CTL into mu calculus formula and then apply the labelling algorithm to find their extensions.

AF (EG (a implies EX AX b))

alpha = AX b = [next] b

beta = EX alpha

gamma = a implies beta

delta = EG gamma

sigma = AF delta

[alpha] = [[next] b] = PreA(next,[b]) = PreA(next,{0,3,4}) = {2,3,4}

[beta] = [EX alpha] = [<next> alpha] = PreE(next,[alpha]) = {0,1,2,3}

[gamma] = [ a implies beta] = [not a or beta] = [not a] union [beta] = {3,4} union {0,1,2,3} = {0,1,2,3,4}

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

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

[X0] = {0,1,2,3,4}

[X1] = [gamma and <next> X0] = [gamma] intersect PreE(next,[X0]) = {0,1,2,3,4} intersect {0,1,2,3,4} = {0,1,2,3,4}

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

[delta] = {0,1,2,3,4}

[sigma] = [AF delta] = [mu X delta or [next] X]

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

[X0] = {}

[X1] = [delta or [next] X0] = [delta] union PreA(next,[X0]) = {0,1,2,3,4} union {} = {0,1,2,3,4}

[X2] = [delta or [next] X1] = [delta] union PreA(next,[X1]) = {0,1,2,3,4} union {0,1,2,3,4} = {0,1,2,3,4}

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

[sigma] = {0,1,2,3,4}

It is 0 in [sigma]? YES, so the CTL formula is true in this transition system

EX4

q1(xr)←e(xr,xg),e(xg,xb),e(xb,xr).

q2(x)←e(x,y),e(y,z),e(z,x),e(z,v)e(v,w),e(w,z).

We want to check if q1 is contained in q2, this means that check if forall x q1(x) implies q2(x) is valid. Validity means that forall I,alpha |= forall x. q1(x) implies q2(x) where I interpretation and alpha assignment.

In FOL validity is undecidable, but with CQ we can make queries satisfiable because we transform them in databases

We need 3 steps:

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

q1(c)←e(c,xg),e(xg,xb),e(xb,c).

q2(c)←e(c,y),e(y,z),e(z,c),e(z,v)e(v,w),e(w,z).

1. Build canonical database corresponding of q1

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

delta^I1 ={c,xg,xb}

e^I1 = {(c,xg),(xg,xb),(xb,c)}

c^I1 = c

1. Check if q2 is true over database Iq1, so this means to verify if there is an assignment forall free variables

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

delta^I2 ={c,y,z,v,w}

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

c^I2 = c

alpha (y) = xg

alpha (z) = xb

alpha (v) = c

alpha (w) = xg

This is a satisfying assignment

Now we need to find an homomorphism. A Homorphism is a mapping between two interpretations, between elements of 2 domains h:delta^I implies 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 the predicate.

Find a homomorphism is to guess a mapping and show that it respects these two properties. But remember theorem that if you have an assignment alpha, which is a satisfying assignment, you can transform alpha in 2 homomorphism between two canonical interpretationsIq1 |= q2 iff h:Iq2->Iq1

(c,y) in e^I2 then (h(c),h(y)) in e^I1 so (c,xg) in e^I1 ok

(y,z) in e^I2 then (h(y),h(z)) in e^I1 so (xg,xb) in e^I1 ok

(z,c) in e^I2 then (h(z),h(c)) in e^I1 so (xb,c) in e^I1 ok

(z,v) in e^I2 then (h(z),h(v)) in e^I1 so (xb,c) in e^I1 ok

(v,w) in e^I2 then (h(v),h(w)) in e^I1 so (c,xg) in e^I1 ok

(w,z) in e^I2 then (h(w),h(z)) in e^I1 so (xg,xb) in e^I1 ok

EX5

Check if the formula phi is valid, means that we want to check that forall interpretation I, I|=phi. Tableaux method is a technique for proving if a formula is satisfiable or not. So this means that to check validity we must prove if I |=phi forall possible interpretation and this is a NP-complete problem.

Check satisfiability means to find an interpretation that satisfied a closed formula, I|=phi. So we need to transform our validity problem in a satisfiable problem by negating the formula phi and check if it satisfiable.

A formula is satisfiable if exist open branch in tableaux, but in our case to test validity we start with not phi and if there are only closed branches the formula is logically valid otherwise not.

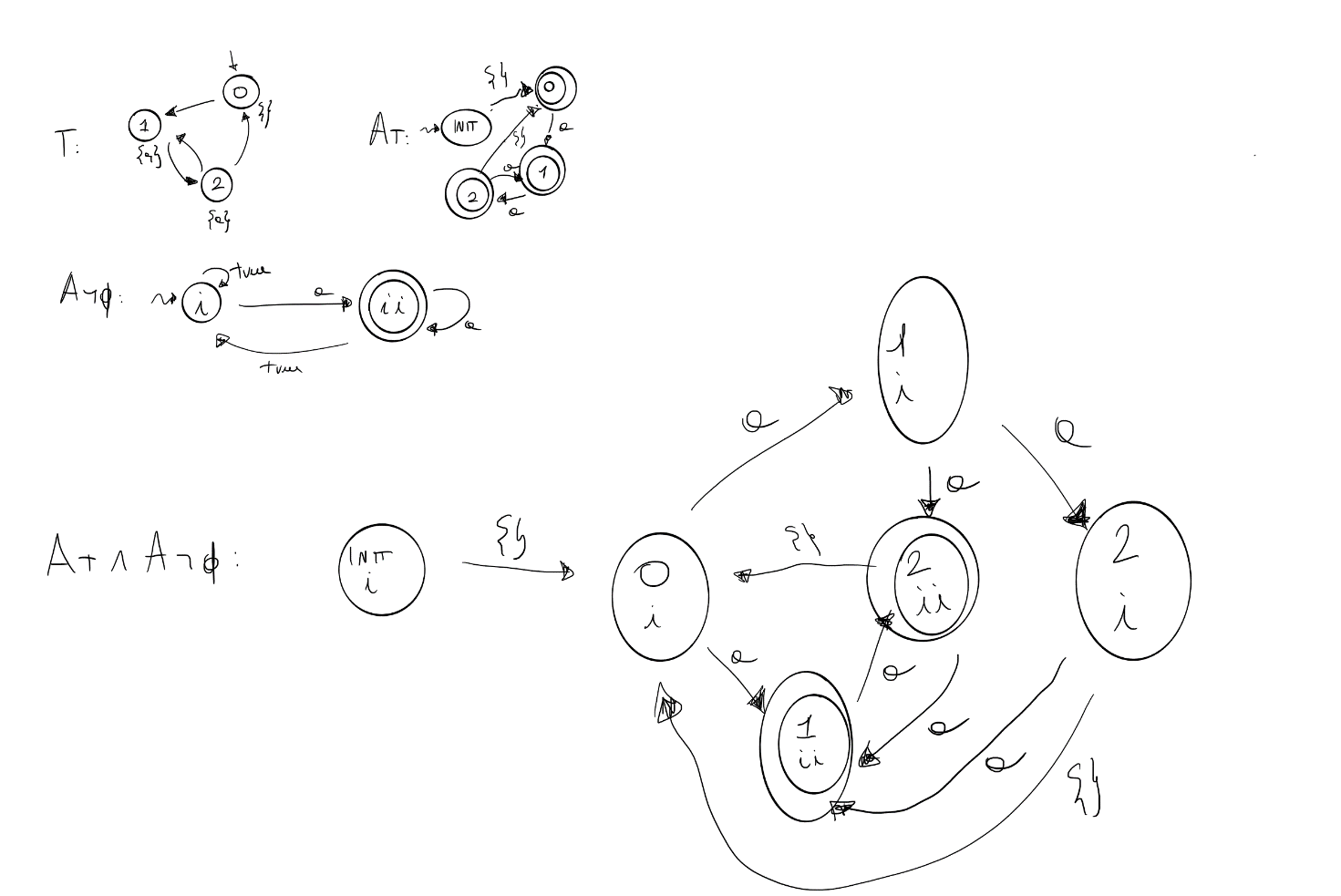
­­Immagine che contiene testo

Descrizione generata automaticamente

EX6

In LTL we cannot do model checking by transforming sub formula into mu calculus formula as CTL and we cannot even exploit NFA or DFA because they work with finite state and they are finite, while LTL is evaluate on infinite languages and has infinite traces. We need to translate LTL formula in NBA that can accept because we can go to a finite state infinite often. To model check LTL formula phi over transition system T , we must prove L(T) subseteq L(phi) iff L(T) and L( no phi) = {}. If we translate T in the automata At and not phi in Anotphi, then we need to show nonemptiness, L(At and Anotphi) ={}.

This is a great solution because using NBA intersection is easier than complement, so in this way we can prove that phi satisfied T, so check if the new automata accept at least a word ( there exists a trace of transition system that satisfied not phi such that from the initial state got to final one and loops there). To prove non emptiness we need to check if At and Anopho,init|=sigma = vXmuY (final and <next>X) or <next>Y



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

We are going to find GFP because of presence of vX

[X0] = {(init,i),(0,i),(1,i),(2,i),(1,ii),(2,ii)}

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

We are going to find LFP because of presence of muY

[Y00] = {}

[Y01] =[(final and <next>X0) or <next>Y00] = ([final] intersect PreE(next,[X0])) union PreE(next,[Y00]) = ({(1,ii),(2,ii)} intersect {(init,i),(0,i),(1,i),(2,i),(1,ii),(2,ii)}) union {} = {(1,ii),(2,ii)}

[Y02] =[(final and <next>X0) or <next>Y01] = ([final] intersect PreE(next,[X0])) union PreE(next,[Y01]) = ({(1,ii),(2,ii)} intersect {(init,i),(0,i),(1,i),(2,i),(1,ii),(2,ii)}) union {(0,i),(1,i),(2,i),(1,ii),(2,ii) } = {(0,i),(1,i),(2,i),(1,ii),(2,ii) }

[Y03] =[(final and <next>X0) or <next>Y02] = ([final] intersect PreE(next,[X0])) union PreE(next,[Y02]) = ({(1,ii),(2,ii)} intersect {(init,i),(0,i),(1,i),(2,i),(1,ii),(2,ii)}) union {(init,i),(0,i),(1,i),(2,i),(1,ii),(2,ii)}

= {(init,i),(0,i),(1,ii),(2,ii)}

[Y04] =[(final and <next>X0) or <next>Y03] = ([final] intersect PreE(next,[X0])) union PreE(next,[Y03]) = ({(1,ii),(2,ii)} intersect {(init,i),(0,i),(1,i),(2,i),(1,ii),(2,ii)})union {(init,i),(0,i),(1,i),(2,i),(1,ii),(2,ii)}= {(init,i),(0,i),(1,i),(2,i),(1,ii),(2,ii)}

[Y04] = [Y03] - - > found LFP

[X1] = {(init,i),(0,i),(1,i),(2,i),(1,ii),(2,ii)}

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

[sigma] = {(init,i),(0,i),(1,i),(2,i),(1,ii),(2,ii)}

IT is (init,i) in [sigma] ? YES, so the LTL formula is true in this transition system, so exist a trace that start from initial state and go to final and loops often.