21-12-17A

EX1

Alphabet:

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

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

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 there is an ISA on it, so we are going to follow this procedure to make instantiation complete

Given a UML diagram T

Iold =0, Inew=I

While (Iold and Inew are different) do

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, BC^I, C^I, contract^I, cost^I, P^I, S^I, provides^I)

I0:

BC^I = {b1,b2,b3}

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

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

P^I = {p1,p2}

S^I = {s1,s2,s3}

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

I1:

BC^I = {b1,b2,b3}

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

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

P^I = {p1,p2}

S^I = {s1,s2,s3}

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

I2:

BC^I = {b1,b2,b3}

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

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

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 are going to check if the instantiation is correct, so if I|=phi. This means that we want to verify if all axioms in T are evaluate true over interpretation I.

Each instance of BusinessCustumer is also an instance of Custumer, ok. For each pair (instance of Custumer and instance of Service) there is only one instance of Provider in that contract, true. Each Provider provides at least 1 and at most 10 service. Each instance of Service is provides by at least 1 Provider.

The instantiation is correct.

1. q() <- Forall x,y. P(x) and S(y) and (Exists z. C(z) and contract(z,y,x)) implies not provides (x,y)

q():false

1. q(x) <- C(x) and Forall y. provides(p1,y) implies (Exists z. P(z) and contract(x,y,z))

q(x):{c1,b2}

1. q(x) <- C(x) and Forall y. S(y) implies (Exists z. P(z) and contract(x,y,z) )

q(x):{c1}

EX3

Model checking a 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 to 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 where each element satisfies phi. To compute this subset we need to apply the labelling algorithm that consists in labelling states of T with predicate that are true in them. The extensions of least fixpoint and greatest fixpoint are computed using Tarski-Knaster approximates theorem.

vX.muY.((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] intersect PreE(next,[Y00])) = ({1,2} intersect {0,1,2,3,4}) union ({3,4} intersect {}) = {1,2}

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

[Y02] = [Y01] - - > found LFP

[X1] = {1,2}

[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] intersect PreE(next,[Y10])) = ({1,2} intersect {1}) union ({3,4} intersect {}) = {1}

[Y12] = [(a and [next] X1) or (b and <next>Y11)] = [([a] intersect PreA(next,[X1])) union ([b] intersect PreE(next,[Y11])) = ({1,2} intersect {1}) union ({3,4} intersect {0}) = {1}

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

[X2] = {1}

[X3] = [muY (a and [next] X2) 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] intersect PreE(next,[Y20])) = ({1,2} intersect {}) union ({3,4} intersect {}) = {}

[Y21] = [Y20] - - > found LFP

[X3] = {}

[X4] = [muY (a and [next] X4) 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,[X3])) union ([b] intersect PreE(next,[Y30])) = ({1,2} intersect {}) union ({3,4} intersect {}) = {}

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

[X4] = {}

[X4] = [X3] - - > found GFP

[phi] = {}

It is 0 in [phi]? No, the formula phi is not satisfied by this transition system

Now we are going to do model checking of CTL formula. Given a Kripke model (KM) = <S,I,R,AP,L> (S is set of state, I set of initial states, R set of transitions, AP set of atomic propositions 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 where each element satisfies phi. 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 we apply the labelling algorithm to fin their extensions.

AF (EG (a implies AX EX not a))

alpha = EX not a = <next> not a

beta = AX alpha

gamma = a implies beta

delta = EG gamma

sigma = AF delta

[alpha] = [<next> not a] = PreE(next,[not a]) = PreE(next,{0,3,4}) = {0,2,3,4}

[beta] = [[next] alpha] = PreA(next,[alpha]) = {1,2,3,4}

[gamma] = [a implies beta] = [not a or beta] = [not a] union [beta] = {0,3,4} union {1,2,3,4} = {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] = [muX 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, the CTL formula is true in this transition system

EX4

Check if Hoare triple {P} while g do S {Q} is correct cannot be done automatically but we need inference rule.

Assuming to have a candidate invariant I, we 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 respect this three operations it is called invariant.

If I have a candidate these operations are automatic, otherwise it is not easy to manage with it. The problem of check if Hoare triple is correct or not is undecidable because we don’t have a sound or complete technique to do this or to find all possible invariants. The only thing that we can do is to check if a candidate invariant is invariant or not.

Candidate: {x>=0 and y>=0 and x+y = 23}

{x= 23 and y= 0}while(x>0) do (x=x−1; y:= y+1){y= 23}

* P implies I

{x= 23 and y= 0} implies {x>=0 and y>=0 and x+y = 23}

Yes, because if x=23 and y=0 means that x>=0 and also y, but in particular that x+y is always equal to 23

* {I and not g} implies {Q}

{x>=0 and y>=0 and x+y = 23 and x<=0} implies {y=23}

{x=0 and y>=0 and x+y = 23} implies {y=23}

It is correct because id x=0 and x+y should be = 23 we mush have y=23

* Compute wp(S,I)

{x>=1 and y>=-1 and x+y = 23} is wp

x = x-1

{x>=0 and y>=-1 and x+y+1 = 23}

y=y+1

{x>=0 and y>=0 and x+y = 23}

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

{x>=0 and y>=0 and x+y = 23 and x>0} implies {x>=1 and y>=-1 and x+y = 23}

{x>0 and y>=0} implies { x>=1 and y>=-1}

The condition is always true, so this means that

{x>=0 and y>=0 and x+y = 23} is invariant

EX5

Check if formula phi is valid means that for all interpretation I, I|=phi. Tableaux is a method for proving, in a mechanical way, if a formula is satisfiable or not. To check validity we must prove that I|=phi, for all possible interpretation and this is a NP-complete problem.

Check satisfiability of a closed formula phi means that exists an interpretation I such that I|=phi. So we need to transform our problem of validity to a problem of satisfiability by negating the formula phi and check if it is satisfiable.

A formula is satisfiable if there is at least one open branch in the tableaux, but in our case we need to have all branches closed because we start with not phi, so if it is the case our formula is logically valid.

Immagine che contiene testo

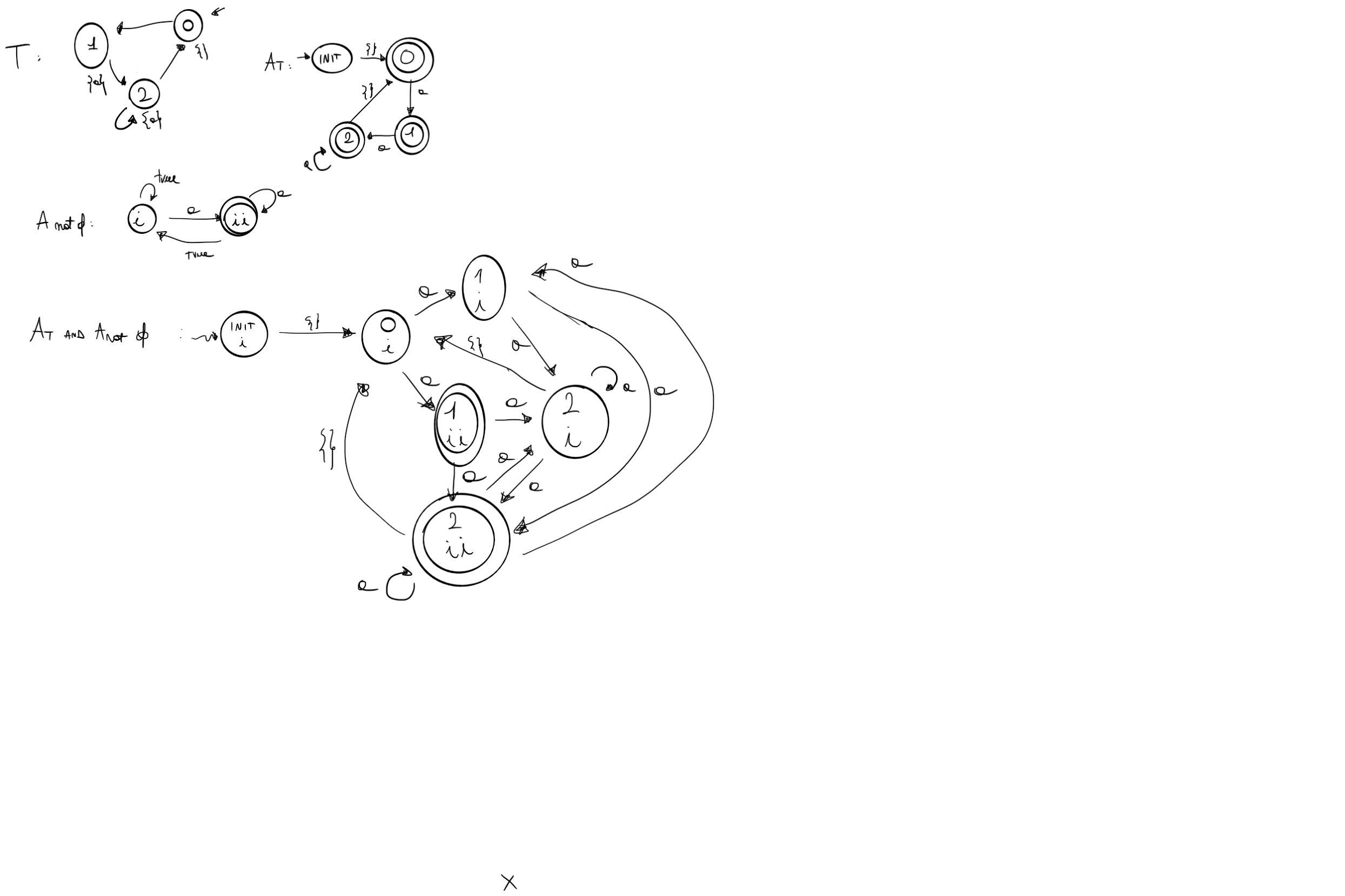
Descrizione generata automaticamente

Formula is valid

EX6

To do model checking with LTL formula, we cannot translate formula in mu calculus formula as CTL. We cannot even exploit in NFA or DFA because they work on finite state while LTL is evaluated in infinite states and has infinite traces. To make Model checking with LTL we can translate formula in NBA because it work with finite states but it goes to finite states infinitely often. Model checking a LTL formula phi over transition system T we need to prove L(T) subseteq L(phi) = L(T) and L(notphi) = {}

We transform T in automata At and notphi in automata Anotphi and then we check the nonemptyness L(At and Anotphi) = {}. In this way we can check in the new automata if LTL formula is satisfied by transition system T, and so we need to check this mu calculus formula At and Anotphi, init|=sigma = vXmuY (final and <next>X) or <next>Y.



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

We are going to find the greatest fixpoint (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 the least fixpoint (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,i),(2,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, the LTL formula is satisfied by this transition system