17-06-19

EX1

Alphabet:

C(x), P(x), S(x), SS(x), contract(x,y,z), cost(x,y,z,k), real(x), specializedin(x,y)

Axioms:

Forall x. SS(x) implies S(x) \\ ISA

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

Forall x. P(x) implies 1<=#{y|specializedin(x,y)} <=5 \\ multiplicity

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

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

EX2

1. The instantiation is not complete because there is an ISA in the UML diagram.

We are going to follow this procedure to complete the instantation

Given a UML diagram T

Iold = {}, 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 of constraints forall x,y. P(x,y) implies R(x,y)

I=Inew

Return I

I = (Obj^I, C^I, P^I, S^I, SS^I, contract^, cost^I, real^I, specializedin^I)

I0:

Obj^I = {c1,c2,c3,c4,s1,s2,s3,ss1,ss2,p1,p2}

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

S^I= {s1,s2,s3}

SS^I = {ss1,ss2}

P^I = {p1,p2}

specializedin^I ={(p1,s1),(p1,s2),(p1,s3),(p2,ss1),(p2,ss2)}

contact^I/cost^I = {(c1,p1,s1,90),(c1,p2,s2,80),(c2,p1,s1,50),(c3,p2,ss1,170),(c2,p2,ss2,100)}

I1:

We are going to put elements of SS in S because of ISA

Obj^I = {c1,c2,c3,c4,s1,s2,s3,ss1,ss2,p1,p2}

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

S^I= {s1,s2,s3, ss1,ss2}

SS^I = {ss1,ss2}

P^I = {p1,p2}

specializedin^I ={(p1,s1),(p1,s2),(p1,s3),(p2,ss1),(p2,ss2)}

contact^I/cost^I = {(c1,p1,s1,90),(c1,p2,s2,80),(c2,p1,s1,50),(c3,p2,ss1,170),(c2,p2,ss2,100)}

I2:

Nothing changes

Obj^I = {c1,c2,c3,c4,s1,s2,s3,ss1,ss2,p1,p2}

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

S^I= {s1,s2,s3, ss1,ss2}

SS^I = {ss1,ss2}

P^I = {p1,p2}

specializedin^I ={(p1,s1),(p1,s2),(p1,s3),(p2,ss1),(p2,ss2)}

contact^I/cost^I = {(c1,p1,s1,90),(c1,p2,s2,80),(c2,p1,s1,50),(c3,p2,ss1,170),(c2,p2,ss2,100)}

I2 = I1

The instantiation is complete

Now we are going to check If the instantiation is correct, so if I|=T. Means that we want to verify if all axioms of T are evaluated true in I.

Each provider should be specializedin at least 1 and at most 5 services, correct.

Each instance of SpecializedService is also instance of Service, ok. For each pairs (custumer, provider) we should have one and only one service in contract, this is true.

The instantiation is correct.

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

q(x):{p1,p2}

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

q(x):{p1}

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

q(x):{p2}

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

q():false

EX3

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

vXmuY (a and <next>X) or (not b<next> Y)

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

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

[X1] = [muY (a and <next>X0) or (not b<next> Y)]

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

[Y00] = {}

[Y01] =[(a and <next>X0) or (not b<next> Y00)] = ([a] intersect PreE(next,[X0])) union ([not b] intersect PreE(next,[Y00])) = ({2} intersect {1,2,3,4}) union ({1,2,3} intersect {}) = {2}

[Y02] =[(a and <next>X0) or (not b<next> Y01)] = ([a] intersect PreE(next,[X0])) union ([not b] intersect PreE(next,[Y01])) = ({2} intersect {1,2,3,4}) union ({1,2,3} intersect {1}) = {1,2}

[Y03] =[(a and <next>X0) or (not b<next> Y02)] = ([a] intersect PreE(next,[X0])) union ([not b] intersect PreE(next,[Y02])) = ({2} intersect {1,2,3,4}) union ({1,2,3} intersect {1,3,4}) = {1,2,3}

[Y04] =[(a and <next>X0) or (not b<next> Y03)] = ([a] intersect PreE(next,[X0])) union ([not b] intersect PreE(next,[Y03])) = ({2} intersect {1,2,3,4}) union ({1,2,3} intersect {1,2,3,4}) = {1,2,3}

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

[X1] = {1,2,3}

[X2] = [muY (a and <next>X1) or (not b<next> Y)]

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

[Y10] = {}

[Y11] =[(a and <next>X1) or (not b<next> Y10)] = ([a] intersect PreE(next,[X1])) union ([not b] intersect PreE(next,[Y10])) = ({2} intersect {1,2,3,4}) union ({1,2,3} intersect {}) = {2}

[Y12] =[(a and <next>X1) or (not b<next> Y11)] = ([a] intersect PreE(next,[X1])) union ([not b] intersect PreE(next,[Y11])) = ({2} intersect {1,2,3,4}) union ({1,2,3} intersect {1}) = {1,2}

[Y13] =[(a and <next>X1) or (not b<next> Y12)] = ([a] intersect PreE(next,[X1])) union ([not b] intersect PreE(next,[Y12])) = ({2} intersect {1,2,3,4}) union ({1,2,3} intersect {1,3,4}) = {1,2,3}

[Y14] =[(a and <next>X1) or (not b<next> Y13)] = ([a] intersect PreE(next,[X1])) union ([not b] intersect PreE(next,[Y13])) = ({2} intersect {1,2,3,4}) union ({1,2,3} intersect {1,2,3,4}) = {1,2,3}

[Y14] = [Y13] - - > found LFP

[X2] = {1,2,3}

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

[phi] = {1,2,3}

It is 1 in [phi]? Yes, so the formula is 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 is set of initial states, R is set of transitions, AP set of atomic proposition and L labelling function) and a CTL formula phi we want to verify that KM,s|=phi where s is state of S. As we can see before with model checking we return a subset of S, in this case to compute it we need to exploit the syntactic structure of CTL formula. In particular, we translate each CTL sub formula in mu calculus formula and then we apply 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 = muX b or <next> X

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

delta = gamma and beta and alpha

sigma = AG delta

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

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

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

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

[alpha] = {1,2,3}

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

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

[X0] = {}

[X1] = [b or <next> X0] = [b] union PreE(next,[X0]) = {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] - - > found LFP

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

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

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

[X0] = {}

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

[X4] = [a or [next] X3] = [a] union PreA(next,[X3]) = {2} union {1,3,4} = {1,2,3,4}

[X5] = [a or [next] X4] = [a] union PreA(next,[X4]) = {2} union {1,2,3,4} = {1,2,3,4}

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

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

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

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

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

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

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

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

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

It is 1 in [sigma]? Yes, the CTL formula is true in this transition system

EX4

q1()←edge(r,g),edge(g,b),edge(b,r)

q2()←edge(x,y),edge(y,z),edge(z,x),edge(z,v),edge(v,w),edge(w,z)

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

In FOL validity is undecidable but with conjunctive queries we can make them satisfiable because we transforms queries in databases

We need 3 steps:

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

In this case we don’t need to do this step because our queries are already Boolean

1. Build canonical database corresponds of q1

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

delta^I1 = {r,g,b}

e^I1 = {(r,g),(g,b),(b,r)}

c^I1 = {}

1. We need to check that q2 is true over database of q1, i.e. we want to find an assignment forall free variables

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

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

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

c^I2 = {}

alpha(x) = r

alpha(y) = g

alpha(z) = b

alpha(v) = r

alpha(w) = g

This is a satisfying assignment.

Now we need to find a homomorphism. A homomorphism is a mapping between two interpretations, 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

We need to guess a homomorphism such that these two properties are satisfied. But we remember that there is a theorem that says that if we have an assignment alpha, a satisfying assignment, we can transform alpha in two homomorphism such that Iq1|=q2 iff Iq2 implies Iq1

The first property is respect, because we don’t have constants.

We show the second:

(x,y) in e^I2 then (h(x), h(y)) is in e^I1? Yes, (r,b) is in e^I1

(y,z) in e^I2 then (h(y), h(z)) is in e^I1? Yes, (b,g) is in e^I1

(z,x) in e^I2 then (h(z), h(x)) is in e^I1? Yes, (g,r) is in e^I1

(z,v) in e^I2 then (h(z), h(v)) is in e^I1? Yes, (r,b) is in e^I1

(v,w) in e^I2 then (h(v), h(w)) is in e^I1? Yes, (b,g) is in e^I1

(w,z) in e^I2 then (h(w), h(z)) is in e^I1? Yes, (g,b) is in e^I1

Ok

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 satisfied 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 consist of find if exist an interpretation I, such that I|=phi. So we are going to transform our problem of validity in a problem of satisfiability by negating the formula and check if it is satisfiable.

A formula is satisfiable if exists at least one open branch. In our case we negate the formula, so we start with not phi, this means that if we have all closed branches our formula is logically valid.

Immagine che contiene testo

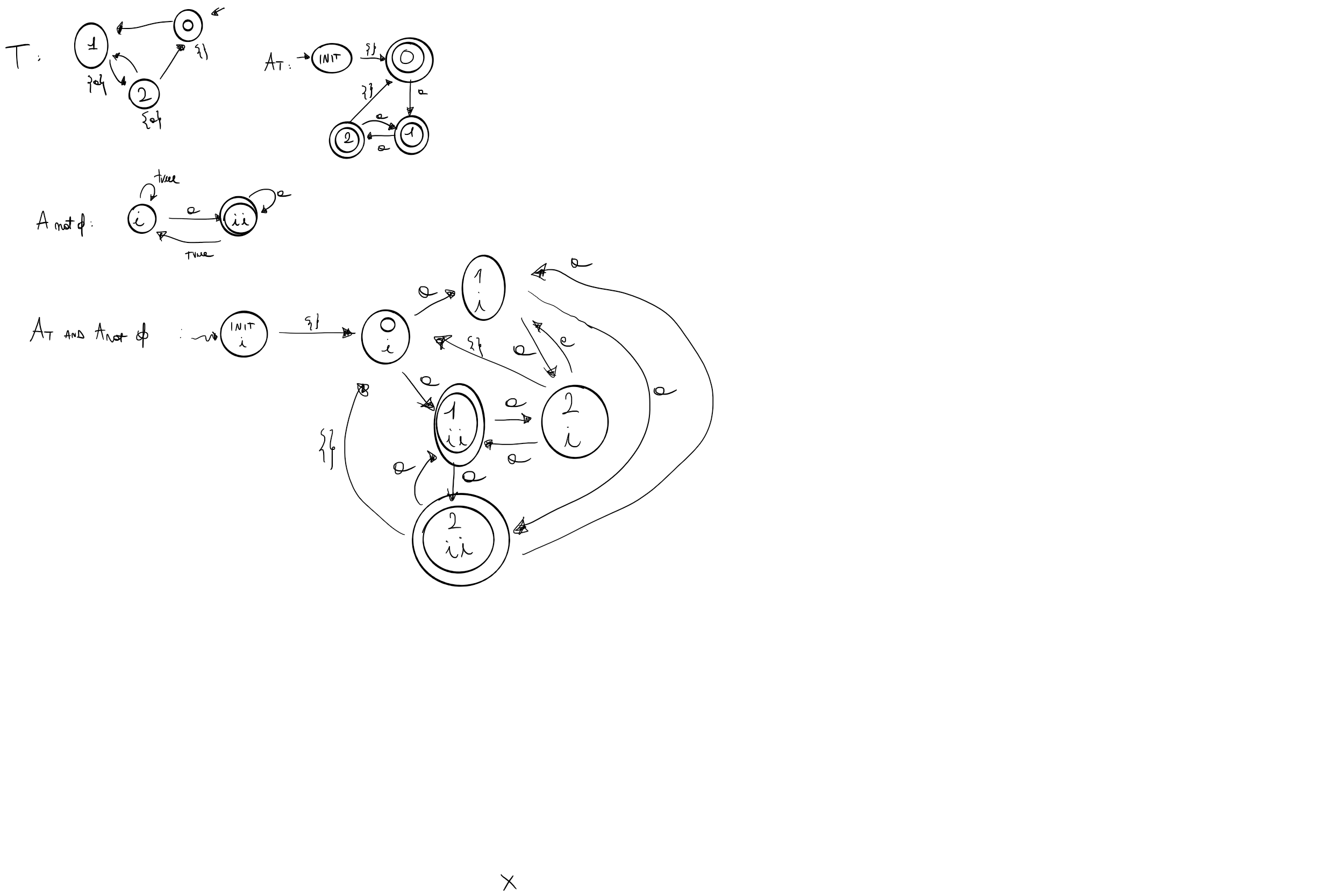
Descrizione generata automaticamente

Formula is valid.

EX6

To do model checking LTL formula phi we cannot translate formula as CTL. We cannot even exploit in NFA or DFA because they work on finite states while LTL work with infinite states and it has infinite traces. The only thing we can do is to transform LTL formula phi in NBA because NBA work with finite state but we go to a final state infinitely often. Model checking of LTL formula phi over transition system T we verify that L(T) subseteq L(phi) iff L(T) and L(notphi) = {}. We transform T in automata At and notphi in automata Anotphi, and we show the nonemptyness L(At and Anotphi) = {}.

In this way we can check if LTL formula phi is satisfied by transition system T, in particular in the new automata At and Anotphi we need to verify that 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 the 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 the 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),(2,ii)} = {(0,i),(1,i),(2,i),(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),(2,ii)} = {(init,i),(0,i),(1,i),(2,i),(2,ii)}

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

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

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

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

It is (init,i) in [sigma]? Yes, so the LTL formula is satisfied by T