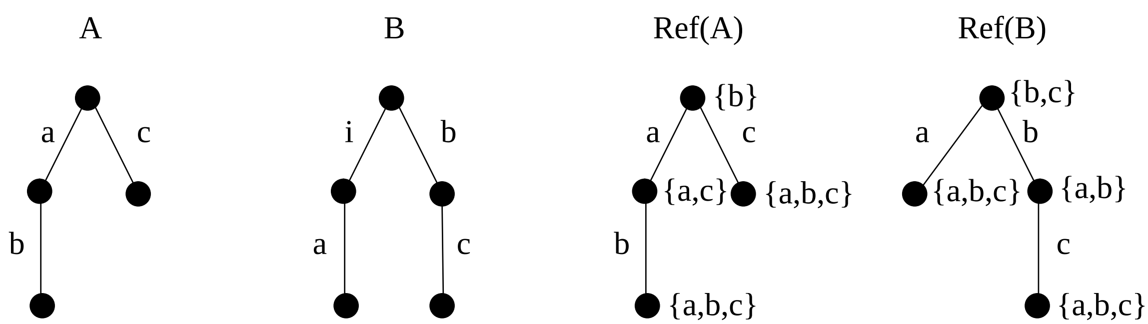
CSI 5109 Assignment 4

1. By constructing the refusal trees of the two behaviour expressions below, show that conformance is not a symmetric relation.

A = a; b; stop [] c; stop
B = i; a; stop [] b; c; stop



Knowing that: I conf S $=_{def}$ $\forall\ \sigma \in \mathrm{Tr}(I) \cap \mathrm{Tr}(S), \mathrm{Ref}(I,\sigma) \subseteq \mathrm{Ref}(S,\sigma)$

Tr(A) = {ε, a, ab, c}
Tr(B) = {ε, a, b, bc}

Tr(A) ∩ Tr(B) = {ε,a}

Ref(A,ε) = {b}
Ref(B,ε) = {b,c}

Ref(A,ε) ⊆ Ref(B,ε)

Ref(A,a) = {a,c}
Ref(B,a) = {a,b,c}

Ref(A,a) ⊆ Ref(B,a)

A conf B since **Ref(A,ε)** ⊆ **Ref(B,ε)** and **Ref(A,a)** ⊆ **Ref(B,a)**; but B does not conform to A since **Ref(B,ε)** ⊄ **Ref(A,ε)** (not to mention that **Ref(B,a)** ⊄ **Ref(A,a)** either).
=> CONF is NOT a symetric relation.

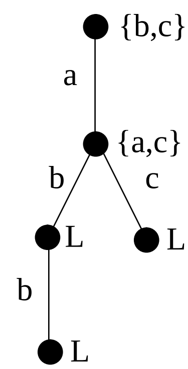
2. Given the behaviour expressions:
A = (a; (b; stop [] i; c; stop)) |[a,c]| (a; (i; b; stop [] c; stop))
B = (a; (b; stop [] c; stop)) |[a, c]| (a; (i; b; stop [] i; c; stop))

a) Are A and B weak bisimulation equivalent?

If A and B are derived on a:
A -a-> A’ = (b; stop [] i; c; stop) |[a,c]| (i; b; stop [] c; stop)
B -a-> B’ = (b; stop [] c; stop) |[a, c]| (i; b; stop [] i; c; stop)
Then A’ (executing ***i*** on left side) and B’ (executing first ***i*** on right side) on ε:
A’ -ε-> A’’ = (c; stop) |[a,c]| (i; b; stop [] c; stop)
B’ -ε-> B’’ = (b; stop [] c; stop) |[a, c]| (b; stop)
The resulting A’’ can execute ***c*** but B’’ cannot => A is NOT weak bisimilar to B.

b) Does one of them conform to the other?

Ref(A) = Ref(B) =



with L = {a,b,c}

Tr(A) = Tr(B) = Tr(A) ∩ Tr(B) = {ε, a, ab, abb, ac}

Ref(A,ε) = Ref(B,ε) = {b,c} => Ref(A,ε) ⊆ Ref(B,ε) and Ref(B,ε) ⊆ Ref(A,ε)
Ref(A,a) = Ref(B,a) = {a,c} => Ref(A,a) ⊆ Ref(B,a) and Ref(B,a) ⊆ Ref(A,a)
Ref(A,ab) = Ref(B,ab) = {a,b,c} => Ref(A,ab) ⊆ Ref(B,ab) and Ref(B,ab) ⊆ Ref(A,ab)
Ref(A,abb) = Ref(B,abb) = {a,b,c} => Ref(A,abb) ⊆ Ref(B,abb) and Ref(B,abb) ⊆ Ref(A,abb)
Ref(A,ac) = Ref(B,ac) = {a,b,c} => Ref(A,ac) ⊆ Ref(B,ac) and Ref(B,ac) ⊆ Ref(A,ac)

Thus, A conf B and B conf A.

c) Are they trace equivalent?

Yes. As shown above: Tr(A) = Tr(B) = {ε, a, ab, abb, ac}

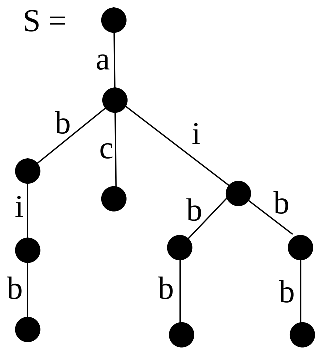
d) Are they testing equivalent?

Since A conf B and B conf A and they share the same traces, A te B.

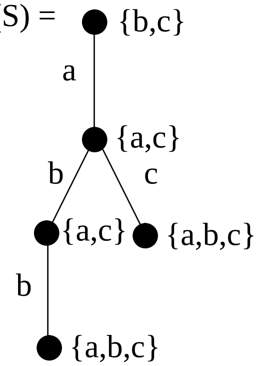
3. Construct the canonical tester of the following behaviour expression and derive the set of test cases:

(a; (b; stop [] c; stop)) |[a,c,]| (a; (i; b; stop [] c; stop))

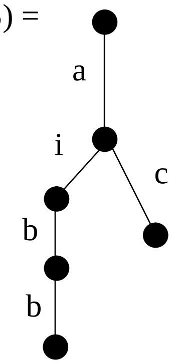
S =



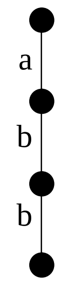
Ref(S) =



T(S) =



Reduction:



L = {a,b,c}
L* = {{},{a},{b},{c},{a,b},{a,c},{b,c},{a,b,c}}

Tr(S) = {ε, a, ab, abb, ac}

Ref(S,ε) = {{},{b},{c},{b,c}}
Ref(S,a) = {{},{a},{c},{a,c}}
Ref(S,ab) = {{},{a},{c},{a,c}}
Ref(S,abb) = L*
Ref(S,ac) = L*

Ref(T(S),ε) = {{},{b},{c},{b,c}}
where $\forall\sigma \in$ Ref(T(S),ε), {a,b,c} $\in$ Ref(S,ε) $\Leftrightarrow$ {a,b,c}\σ $\in$ Ref(S,ε)

Ref(T(S),a) = {{},{a},{c},{a,c}}
where $\forall\sigma \in$ Ref(T(S),a), {a,b,c} $\in$ Ref(S,a) $\Leftrightarrow$ {a,b,c}\σ $\in$ Ref(S,a)

Ref(T(S),ab) = {{},{a},{c},{a,c}}
where $\forall\sigma \in$ Ref(T(S),ab), {a,b,c} $\in$ Ref(S,ab) $\Leftrightarrow$ {a,b,c}\σ $\in$ Ref(S,ab)

Ref(T(S),abb) = L*
where $\forall\sigma \in$ Ref(T(S),abb), {a,b,c} $\in$ Ref(S,abb) $\Leftrightarrow$ {a,b,c}\σ $\in$ Ref(S,abb)

Ref(T(S),ac) = L*
where $\forall\sigma \in$ Ref(T(S),ac), {a,b,c} $\in$ Ref(S,ac) $\Leftrightarrow$ {a,b,c}\σ $\in$ Ref(S,ac)

4. By reference to the notes by Burstall, prove the following:

a) For all m, n, m + succ(n) = succ (m+n) [Proposition 5.2 on page 7]

lemmas:
0 + n = n (1)
succ(m) + n = succ(m + n) (2)
I.H.
m + succ(n) = succ(m + n)

Base: m = 0
0 + succ(n) = succ(0 + n)
LHS reduced by lemma (1):
succ(n) = succ(0 + n)
RHS reduced by lemma (1):
succ(n) = succ(n)

Step: m -> succ(m)
succ(m) + succ(n) = succ(succ(m) + n)
LHS by lemma (2):
succ(m + succ(n)) = succ(succ(m) + n)
LHS by I.H.:
succ(succ(m + n)) = succ(succ(m) + n)
RHS by lemma (2):
succ(succ(m + n)) = succ(succ(m + n))

b) For all l, join(l,nil) = l [first Lemma 6.1 on page 9]

lemmas:
join(nil,l) = l (1)
join(s::k,l) = s::join(k,l) (2)
I.H.:
join(l,nil) = l

Base: l = nil
join(nil,nil) = nil
LHS by lemma (1):
nil = nil

Step: l -> s::l
join(s::l,nil) = s::l
LHS by lemma (2):
s::join(l,nil) = s::l
LHS by I.H:
s::l = s::l

c) Given the definition:

length : list(alpha) -> nat

length(nil) <= 0 (1)

length(n::l) <= length(l) + 1 (2)

Prove that:

length(join(k,l)) = length(k) + length(l)

lemmas:

0 + n = n (3)

join(nil,l) = l (4)

join(s::k,l) = s::join(k,l) (5)

∀ m,n ∈ nat, m + n = n + m (6)

I.H.

length(join(k,l)) = length(k) + length(l)

Base: k = nil

length(join(nil,l)) = length(nil) + length(l)

RHS by (1):

length(join(nil,l)) = 0 + length(l)

RHS by lemma (3):

length(join(nil,l)) = length(l)

LHS by lemma (4):

length(l) = length(l)

Step: k -> s::k

length(join(s::k,l)) = length(s::k) + length(l)

RHS by (2):

length(join(s::k,l)) = length(k) + 1 + length(l)

LHS by lemma (5):

length(s::join(k,l)) = length(k) + 1 + length(l)

LHS by (2):

length(join(k,l)) + 1 = length(k) + 1 + length(l)

LHS by I.H.:

length(k) + length(l) + 1 = length(k) + 1 + length(l)

LHS by lemma (6):

length(k) + 1 + length(l) = length(k) + 1 + length(l)