We volgen generate & prune.

Generate: **Voeg aan elk netwerk alle mogelijke comparatoren toe die nuttig zijn**. De ‘nuttigheid’ van een comparator wordt weerspiegeld in het al dan niet veranderen van de outputs van het netwerk.

Prune: We elimineren netwerken die gesubsumed worden door andere netwerken.

**Subsume: Ca subsumes Cb ⬄ Er bestaat een permutatie p waarvoor geldt p(outputs(Ca)) ⊆ outputs(Cb) → verwijder Cb.**

Bijgevolg gaan we per netwerk een set bijhouden van al de huidige outputs. Deze partitioneren we naargelang het aantal 1’en in de output.

Aangezien voor elke deelverzameling (outputs met hetzelfde #1’en) geldt dat het aantal outputs niet verandert door een permutatie kunnen we stellen dat als er een deelverzameling is in Ca die groter is dan de overeenkomstige deelverzameling van Cb, er geen permutatie waardoor Ca Cb zou subsumen.

**L4: Noem de verzameling van outputs met k 1’en: OL(k, C). Dan geldt:  
Er bestaat een K waarvoor |OL(K, Ca)| > |OL(K, Cb)| → Ca subsumed Cb niet.**

Per deelverzameling van outputs gaan we bijhouden op welke posities er een 1 kan staan en op welke posities een 0. W(C,x,k) = 00111 vertaald zich naar voor de set van outputs met k 1’en komt het symbool x voor op plaats 3, 4 en 5 en nooit op plaats 1 en 2.

Voor ons voorbeeld wordt dit: (C, x, k)

W(C1, 1, 3) = 01111 W(C1, 0, 3) = 11101

W(C3, 1, 3) = 11111 W(C3, 0, 3) = 11001

Verwisselen van kolommen (permutatie) verandert niet het aantal mogelijke plaatsen van voorkomen. Verwisselen we bv positie 4 en 5 dan wordt W(C3,0,3) = 11010, het aantal posities (3) blijft echter gelijk. Als de outputs van Ca via permutatie p een deelverzameling moeten zijn van de outputs van Cb moet dus voor elke x en k gelden: |W(Ca, x, k)| ≤ |w(Cb, x, k)| zoniet bestaat er geen permutatie waardoor Ca Cb subsumed.

**L5: Er bestaat een K en een symbool x waarvoor geldt: |w(Ca, x, k)| > |w(Cb, x, k)| → Ca subsumed Cb niet.**

We willen dus beslissen of er een permutatie bestaat die outputs(Ca) omzet naar deelverzameling van outputs(Cb). Dus ook voor elke partitie van Ca (op # 1’en) moet gelden dat deze deelverzameling is van de partitie bij Cb. We kunnen mogelijke permutaties voorstellen door voor elke positie een lijst van getallen. Bv

0 0 3 4 2  
1 1 4 3  
 2   
 3

Zorgt voor de mogelijke permutaties: 01342 en 10342 (de laatste 3 kunnen enkel 342 als geldige sequentie hebben).

Op deze manier zorgt

0 0 0 0  
1 1 1 1  
2 2 2 2  
3 3 3 3

Voor elke permutatie van lengte 4. Alle mogelijke permutaties afgaan is kostelijk, bijgevolg zoeken we een reductie van alle mogelijkheden.

Als p(outputs(Ca)) ⊆ outputs(Cb) moet dus gelden (voor alle x en k) dat w(Ca, x, k) ⊆ w(Cb, x, k) want p(outputs(Ca)) bevat enkel outputs die Cb ook bevat.

Stel dus dat voor een gegeven x en k Cb een 1 heeft op een positie, dit wilt zeggen dat er een output is bij Cb die x heeft op die positie. Voor Ca wilt dit zeggen dat die zowel een 1 als een 0 mag hebben op deze positie, afhangende van of Ca de output van Cb ook bevat. *Als Cb echter een 0 heeft op een positie weten we dat p(Ca) ook op die positie een 0 moet hebben, anders zou p(Ca) een output hebben die geen deel is van outputs(Cb).*

Stel dus bv w(Ca, 1, k) = 10010 en w(Cb, 1, k) = 10110

De posities waar Ca nooit een 1 heeft is op positie 1,2 en 4. We weten dat op positie 1 en 4 nooit een 0 mag komen. Bijgevolg kunnen we alle mogelijke permutaties beperken tot deze waarbij op positie 1 en 4 enkel 1,2 of 4 kan staan.

0 1 0 0 1  
1 2 1 1 2  
2 4 2 2 4  
3 3 3  
4 4 4

Hetzelfde kan men doen voor w(Ca, 0, k) = 01101 met w(Cb, 0, k) = 11011. Ca heeft nooit een 0 op positie 0 en 3. Cb mag nooit een 0 hebben op positie 2. Bijgevolg zal voor positie 2 enkel 0 en 3 volstaan.

0 0 0 0 0  
1 1 3 1 1  
2 2 2 2  
3 3 3 3  
4 4 4 4

*Deze moeten beide gelden.* Bijgevolg nemen we de doorsnede van beide gevonden reducties. (Bij binaire representatie is doorsnede slechts een &-operatie).

0 1 0 0 1  
1 2 3 1 2  
2 4 2 4  
3 3  
4 4

*Dit kunnen we herhalen voor elke k !* Door dit voor elke k en x te herhalen kunnen we een grote reductie bekomen in het aantal mogelijke permutaties. Enkel de overgebleven permutaties moet men nagaan door de outputs van Ca te permuteren en na te gaan of er 1 bestaat waarvoor alle outputs gepermuteerd worden naar deelelementen van Cb.

Van zodra er op een positie geen enkele mogelijkheden zijn kunnen we dus meteen stoppen en vast stellen dat er geen permutatie bestaat die aan de voorwaarde voldoet.

**L6: p(outputs(Ca)) ⊆ outputs(Cb) ⬄ Voor elke x en k geldt p(w(Ca, x, k)) ⊆ w(Cb, x, k)**