2.

2.1  
Bei Vacuity handelt es sich um nichts aussagende/leere Assertions. Ein Beispiel dafür wäre *"always (a -> next b)"*, unter der Annahme das a nie zutrifft. Da a immer falsch ist, ist es irrelevant was auf der rechten Seite steht, man kann den bestehenden Ausdruck durch einen beliebigen boolschen Ausdruck ersetzen.  
Dieses Konzept ist wichtig da damit mögliche Fehler aufgedeckt werden können. Nimmt man das obige Beispiel erwartete sich die Person, die die Assertion schrieb wahrscheinlich das a irgendwann wahr ist.  
Es ist auch für andere Logiken wichtig die obige Bedingung kann man ebenso in LTL/CTL beschreiben.

2.2  
a) x: 0 -> 0 -> 0 -> 0 -> 0 -> 0 …

b) x: 0 -> 0 -> 0 -> 0 -> 0 -> 0 …

c) x: 0 -> 1 -> 1 -> 0 -> 0 -> 0 …  
 y: 0 -> 1 -> 0 -> 0 -> 1 -> 0 …

d) x: 0 -> 1 -> 1 -> 0 -> 1 -> 0 …  
 y: 0 -> 1 -> 0 -> 0 -> 1 -> 0 …

2.3  
Die schwache Version ergibt wenig Sinn, da sie immer wahr sein würde. Denn nach der Simulation könnte die *eventually* Bedingung noch erfüllt werden und im Zweifel ist die schwache Version wahr.

2.4  
Es wird in einem Initialzustand begonnen. Dieser Zustand kommt zu den neuen Zuständen hinzu.  
Danach wird überprüft ob ein neuer Zustand eine Assertion verletzt. Falls eine Assertion verletzt wurde, wird ein Gegenbeispiel ausgegeben. Falls keine verletzt wurde, werden alle nachfolgenden Zustände ermittelt. Dies geschieht indem bei den zuvor überprüften Zuständen alle möglichen Eingabekombinationen angelegt werden. Sind bei den Nachfolgezuständen neue Zustände dabei werden diese wieder überprüft und ihre Nachfolger erzeugt. Sobald keine neuen Zustände gefunden werden, wird der Beweis ausgegeben.

3.

3.1

Anhand des kombinierten Graphen kann man erkennen, dass alle Bedingungen erfüllt werden. Es gibt keinen Zustand C1C2 und nach einem T Zustand kommt immer ein C Zustand.

3.2

Man kann anhand des Zustandsgraphen erkennen das die erste Bedingung nicht mehr erfüllt wird, denn es gibt nun einen Zustand C1C2 indem beide kritischen Zustand sind. Die letzten beiden Bedingungen werden jedoch weiterhin erfüllt.  
Gegenbeispiel: N1N2->(d1=1,d2=2)->T1T2->C1C2->N1N2