Codificación de constraints "at most one" (AMO) en SAT.

En vez de amo(x1...xn) escribiremos: x1+...+xn <= 1.
Nota: aqui x1...xn tambien pueden ser literales negativos (variables negadas).

Codificación	num vars auxiliares	num clausulas
cuadrática	0	n sobre 2
ladder	n	3n
Heule 3	n/2	3n
Heule 4	n/3	3.3n
log	log n	n log n

Cuadrática:
Todas las clausulas binarias de la forma -x_i v -x_j, para 1 <= i < j <= n.
-x_1 v -x_2, -x_1 v -x_3, ..., -x_n-1 v -x_n

Ladder:
Variabes auxiliares a_i que significan: "uno de x_1...x_i es cierto".
Para cada i tenemos clausulas:

x_i -> a_i		-x_i v a_i
a_i -> -x_i+1	que en forma clausal son:	-a_i v -x_i+1
a_i -> a_i+1		-a_i v a_i+1

Heule 3:
Usa la propiedad de que amo(x_1 ... x_n) ssi
amo(x_1, x_2, x_3, aux) AND amo(-aux, x_4 ... x_n).
Después, amo(-aux, x_4 ... x_n), que tiene dos variables menos, se
descompone recursivamente de la misma manera, y amo(x_1, x_2, x_3, aux)
se expresa de la forma cuadrática (6 clausulas).
Así, para quitar dos variables hace falta una variable auxiliar y 6
clausulas. Total: n/2 variables auxiliares y 3n clausulas.

Heule 4:
Como Heule 3, pero con amo(x_1, x_2, x_3, x_4, aux). Para quitar
tres variables hace falta una variable auxiliar y 10 clausulas.
Total: n/3 variables auxiliares y 3.3n clausulas.

Log:
Introducimos log n variables auxiliares b_0...b_k que representarán en
binario la i de la x_i que es cierta. Ejemplo para n=16: habrá 4
variables auxiliares, y por ejemplo, como 6 en binario es 0110, hay
cláusulas:

x_6 -> -b_0		-x_6 v -b_0
x_6 -> b_1	que en forma clausal son:	-x_6 v b_1
x_6 -> b_2		-x_6 v b_2
x_6 -> -b_3		-x_6 v -b_3

Así, con log n clausulas por cada x_i, son n log n clausulas en total.

Codificación de otras restricciones en SAT.

Cardinality constraints: x1+...+xn <= k para algun natural k.

Por ejemplo, "at most 7": x1+...+xn <= 7.
Pero tambien hay "at least", "exactly", etc., es decir, que el operador
puede ser >, <, >=, <=, =.

Codificaciones:

1. Sin variables auxiliares.
Similar a la codificación cuadrática para AMO(x1...xn).
Por ejemplo si k=2 (es decir, el caso x1+...+xn <= 2) tendríamos
un número cúbico de cláusulas de la forma

-x_i v -x_j v -x_k, para 1 <= i < j < k <= n (total: n sobre 3 clausulas).
Similarmente, si k=3 tendríamos O(n^4) cláusulas, de la forma
-x_i v -x_j v -x_k v -x_m, para 1 <= i < j < k < m <= n
(total: n sobre 4 clausulas).

2. Ladder encoding.
Con variables auxiliares, similar a la ladder encoding para AMO(x1...xn).
Por ejemplo si k=2, es decir, el caso x1+...+xn <= 2, tendríamos 2n variables auxiliares
a_1...a_n y b_1...b_n, donde
cada a_i significa: "al menos uno de x_1...x_i es cierto" y
cada b_i significa: "al menos dos de x_1...x_i son ciertos".
Para cada i tenemos clausulas:
 A) -x_i v a_i
 B) -a_i v a_i+1
 C) -b_i v b_i+1
 D) -a_i v -x_i+1 v b_i+1
 E) -b_i v -x_i+1
donde las clausulas de tipo B,C,D,E solo existen para i en 1..n-1. Total: 5n-4 clausulas.
Similarmente podemos tratar el caso de k=3, con 3n variables auxiliares, etc.

3. Sorting networks.
Los cardinality constraints también se pueden codificar en SAT usando
sorting networks. Un sorting network recibe como entrada las
variables x1...xn, y da como salida s1...sn los mismo valores, pero
ordenados. Por ejemplo si la entrada (los valores para x1...x8) es
10110110, la salida s1...s8 será 11111000.

De esta manera podemos expresar, por ejemplo, x1+...+xn <= 6, simplemente
imponiendo que s7=0.
O podemos expresar, por ejemplo, x1+...+xn > 4, poniendo s5=1.
O podemos expresar, por ejemplo, x1+...+xn = 4, poniendo s4=1, y s5=0.

Los sorting networks se construyen mediante "comparadores", ver el
primer dibujo en http://en.wikipedia.org/wiki/Sorting_network, el que habla de
"A simple sorting network consisting of four wires and five connectors".
En nuestra codificación SAT, el valor de cada hilo interno del dibujo
se guarda en una variable auxiliar. Dados dos hilos x,y, el valor del
hilo máximo se obtiene haciendo el OR, y el valor del minimo se
obtiene haciendo el AND.
Expresar A=AND(x,y) se hace con tres cláusulas (como en la
codificacion de Tseitin para circuitos arbitrarios):
  -x v -y v A,   -A v x,   -A v y,   y el OR se hace con tras tres.
Así, como cada comparador necesita seis cláusulas, y el sorting network
necesita n log² n comparadores, en total se necesitan 6 n log² n
clausulas y n log² n variables auxiliares.

Pseudo-Boolean constraints: a1 x1 + ... + an xn <= k para enteros a1,...,an,k.
==========

Por ejemplo, 7x + 8y + 4z <= 11.

Esto se puede codificar en SAT expresando estos constraints como
BDDs (Binary Decision Diagrams), ver la definición y el ejemplo
en http://en.wikipedia.org/wiki/Binary_decision_diagram.
Los BDDs se pueden codificar en SAT con una variable auxiliar A y
cuatro clausulas por cada nodo, de una forma similar a la codificación
de Tseitin:
  -x & -F -> -A
  -x & F -> A
   x & -T -> -A
   x & T -> A
donde F y T son los nodos hijos para false y true.