Lógica en la Informática **Final. jueves 16 de junio 2011**

Corrección/discusión/revisión: martes 28 de junio, 15h. Omega-139
Tiempo: 2h. Sin apuntes.

1 Un conjunto $P = \{p_1 \dots p_n\}$ de personas, con n muy grande, están pensando en reunirse. Hay un subconjunto $S \subset P$ de personas que seguro irán a la reunión. Hay varios subconjuntos $I_1, \dots, I_k$ de P (no necesariamente disjuntos) de personas incompatibles entre sí, es decir, por cada I_i al menos una persona de I_i no va. Finalmente, hay numerosas personas p_i que tienen un grupo de amigos A_i, que dicen: "si van todos mis amigos A_i, entonces yo también". Se trata de decidir si es posible la reunión. ¿Cómo resolverías este problema de manera eficiente con SAT? ¿Qué SAT solver utilizarías?

Solucion: Usamos variables p_i que significan "la persona p_i va a la reunión".
Por cada subconjunto $I_i = \{q_1 \dots q_m\}$ tenemos una cláusula $\neg q_1 \vee \dots \vee \neg q_m$.
Por cada p_i con amigos $A_i = \{q_1 \dots q_m\}$ tenemos una cláusula $\neg q_1 \vee \dots \vee \neg q_m \vee p_i$.
Por cada p_i del conjunto S, tenemos una cláusula unitaria p_i.
Puesto que el problema resultante es Horn, cualquier SAT solver que haga unit propagation lo decide en tiempo polinómico (sin necesidad de backtracking, backjumping, learning, etc.).

2 La reunión de la pregunta anterior fracasó. Por eso las personas de P deciden hacerse llamadas teléfonicas bilaterales (entre dos personas), todos a todos, durante $n-1$ días, de manera que cada día cada uno llama a una persona distinta. Pero vuelven a tener muchas restricciones! Esta vez, todas son de la forma "el dia d, la llamada entre p_i y p_j no puede tener lugar" (con $1 \leq i < j \leq n$ y $1 \leq d < n$). ¿Cómo resolverías este problema de manera eficiente con SAT? ¿Qué SAT solver utilizarías?

Solucion: Usamos variables x_{ijd} que significan "i habla con j el día d" (con $1 \leq i < j \leq n$ y $1 \leq d < n$). Por cada $1 \leq i < j \leq n$ tendremos una cláusula $x_{ij1} \vee \dots \vee x_{ij\,n-1}$ para indicar que cada par de personas (i, j) habla al menos un día. Y hay que expresar que en un mismo día d una persona i no puede tener más de una conversación: por cada $1 \leq i \leq n$ y $1 \leq d < n$, y por cada par $1 \leq j < j' \leq n$ con $i \neq j$ y $i \neq j'$, una cláusula binaria $\neg x_{ijd} \vee \neg x_{ij'd}$. Finalmente, por cada restricción del tipo "el dia d, la llamada entre p_i y p_j no puede tener lugar", una cláusula unitaria $\neg x_{ijd}$. Para este problema (que sí es NP completo) lo mejor es usar un buen SAT solver moderno.

3 Resuelve el problema de la pregunta 2 con Constraint Logic Programing en gprolog.

Solucion:

```
p:-L = [X12, X13, ..., Xn-1 n],       % si Xij=d, llamada i-j (con i<j) es el dia d
   fd_domain(L,1,n-1],
   Xij #\= d                          % para cada restriccion de este tipo
   ...
   fd_all_different(X1i,X2i,...,Xin), % un all_different por cada i con 1 <= i <= n,
   ...                                % con todas las variables Xi* y X*i
   fd_labeling(L), write(L).
```

4 Para hacer más eficiente el algoritmo de la pregunta 3, podríamos haber usado el siguiente tipo de *pairing constraints*. Cada día d, las personas del conjunto $P = \{p_1 \dots p_n\}$ han de formar parejas. Por ejemplo, si ese día a p_i sólo le quedan p_j y p_k para llamar, y a p_j sólo le quedan p_i y p_k, entonces podemos deducir que p_i y p_j deben quedar emparejados (porque si uno de los dos llama a p_k, el otro no tiene a quién llamar). En los siguientes dos grafos, una arista $i - j$ significa que i y j aún pueden quedar emparejados. Indica para cada uno de los dos ejemplos qué aristas se podarían en un algoritmo de propagación arco-consistente para pairing constraints.

Solucion: Se podarían las aristas que no están incluídas en ninguna solución, las indicadas con “x”. Para cada una de las demás aristas sí existe alguna solución que las incluya.