Labos LI 1: Resolución de problemas con SAT

Lo primero que hay que hacer es estudiar muy bien cómo funciona el ejemplo de `miSudoku.pl`. Es un ejemplo de uso de una plantilla prolog que sirve para resolver problemas de todo tipo mediante un SAT solver: permite declarar las variables SAT que tendremos, generar las cláusulas de forma muy cómoda, llama al SAT solver, recupera el model generado por el solver, y mostrar la solución del problema original.

En el `main`, podéis ver que se generan las clausulas (en un archivo `clauses`), y la cabecera, y se genera el `infile.cnf` que es la entrada con la que se llama al SAT solver `picosat`. En el `treatResult`, en caso de satisfactible, a partir del modelo que ha generado `picosat`, se genera lo que llamamos el modelo simbólico `M`: la lista de todos los literales ciertos, en nuestra notación, en forma `x(I,J,K)` (no los números del picosat), y a partir de esa `M` se llama a `displaysol(M)`. Mira cómo funciona para escribir el sudoku resuelto. Prueba ejecutar.
Con esta plantilla prolog, el usuario sólo ha de:

1) declarar las variables SAT que va a tener (ver: `1. SAT Variables`),

2) generar las clausulas programando el predicado `writeClauses` y

3) programar el `displaySol`.

La parte `writeClauses` conviene estructurarla, como se ha hecho en este ejemplo del sudoku. También conviene tener definiciones previas (ver: `Some helpful definitions`), para no ensuciar esta parte. Podeís ver que se usa el predicado que ya os damos hecho `writeClause(Lits)` donde `Lits` es una lista de literales, y que también se facilita una libreria de predicados que generan automáticamente las cláusulas correspondientes a restricciones (constraints) del tipo `exactly(K,Lits)`, `atleast(K,Lits)`, y `atmost(K,Lits)`. Si queréis podéis mirar cómo funciona esta librería. Es interesante, aunque aquí solo vais a ser usuarios.

Problema 1: planificación de ligas deportivas. La liga española de primera división de fútbol tiene 20 equipos, que juegan todos contra todos en 19 jornadas la primera vuelta, y otra vez en otras 19 jornadas en la segunda vuelta, en el mismo orden, pero en casa del otro. Hay que admitir los siguientes tipos de restricciones:

–el equipo i no quiere jugar en casa la jornada j,

–el equipo i sí quiere jugar en casa la jornada j,

–en las jornadas i e $i+1$ no se admiten repeticiones: dos partidos seguidos en casa, o dos partidos seguidos fuera,

–el partido i-i' (es decir, en casa del equipo i) no debe ser la jornada j

–el partido i-i' (es decir, en casa del equipo i) sí debe ser la jornada j.

No se permitirán tripeticiones, es decir, ningún equipo jugará tres jornadas seguidas en casa ni tres jornadas seguidas fuera. Recomendación: primero aborda un subproblema, por ejemplo, una sola vuelta, pocos equipos, etc. Extensión: a lo largo de la temporada ningún equipo tiene más de k repeticiones (esto puede hacer que el problema SAT sea duro para ligas de mas de 16 equipos).

Problema 2: planificación del horario semanal de la FIB: cinco días de 8 a 20h (12 horas diarias). En los datos de entrada se indican las listas de aulas, profesores, y cursos que hay. Todos ellos reciben un natural no nulo como identificador. Por cada curso se da su lista de asignaturas, cada una con su número de horas semanales de clase (máximo una al día), la lista de aulas adecuadas para ella, y la lista de profesores que la podrían impartir. Todas las sesiones de una misma asignatura deben impartirse en la misma aula por el mismo profesor. Por cada profesor se indican las horas (con un entero entre 1 y 60) en que no puede dar clase. Por supuesto, no puede haber más de una clase a la vez por profesor ni por aula. Cada curso ha de tener un horario compacto (sin horas libres entre dos clases el mismo día) y no más de seis horas de clase al día. Sesiones de un mismo curso no pueden solaparse en el tiempo. Ayuda: es mejor introducir diversos tipos de variables que relacionan dos entidades (profesor-asignatura, asignatura-hora, etc.) que tener variables que relacionan más de dos. Hay que mostrar la solución por horas (cada hora qué hay) y despúes de cada curso (qué clases tiene).

Problema 3: Hay que hacer un solver para el juego de Flow (Flow Free) que existe sobre Android, iPhone, etc. Atención: hay que llenar todo el tablero. Proporcionamos un predicado displaySol(M) para este problema. Podéis ver la salida para la primera entrada en el archivo `salidaFlow.pdf`.