KColorSAT
Pràctica de Programació Lògica

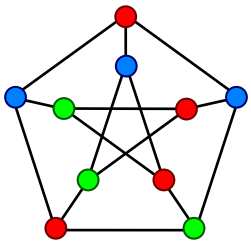
Paradigmes i Llenguatges de Programació

Curs 19/20

Introducció

Degut a les circumstàncies actuals, una universitat s'ha posat en contacte amb nosaltres per tal que l'ajudem a resoldre un problema. Resulta que com que han de fer examens virtuals, han de llogar un software que els permeti "controlar" els alumnes. Aquest software es paga per dies d'ús. Per tant volen fer-ho servir el menor nombre de dies possibles. Ara bé, volen respectar que els alumnes no tinguin més d'un examen al dia. Un recercaire va llegir i comprobar que resoldre aquest problema era el mateix que trobar el *chromatic number* k d'un graf i k-colorejar-lo, és a dir, trobar el nombre mínim de colors necessaris per tal de poder assignar un color a cada node de manera que si dos nodes es toquen (tenen una aresta que els uneix) tinguin colors assignats diferents. La idea és que cada assignatura és una node i si dues assignatures comparteixen algun alumne matriculat llavors els dos nodes corresponents estan conectats amb una aresta. Cada color usat representa un dia utilitzat on es faran els examens de totes les assignatures/nodes amb aquell color.

Pere exemple, aquest graf te nombre cromàtic igual a 3, o el podeu colorejar amb menys de 3 colors?



Què i com ho hem de fer

Com que a classe de PLP hem vist què és això de SAT i resulta que es pot resoldre la k-colorabilitat de grafs codificant aquest problema a SAT, el que farem serà el següent:

- Farem un SAT solver amb Prolog (ja estarà parcialment implementat).
- Farem un programa amb Prolog que, donat un graf G, i un número k, construeixi una fórmula Booleana en forma normal conjuntiva (CNF) que serà satisfactible si i només si el graf G es pot colorejar amb k colors.

- Farem un programa amb Prolog que donat un graf G, anirà preguntant al SAT solver si es pot colorejar amb $1, 2, 3, ...$ colors fins que el SAT solver digui que si. Fixeu-vos que cada pregunta significa construir una CNF nova.

- Un cop haguem trobat que es pot, utilitzarem el model de la fórmula per a mostrar quin color te cada node (quin dia es farà l'examen de cada assignatura).

Tot seguit donem més detalls de com fer cada part.

SAT solver

El primer que farem serà implementar el predicat `sat(...)` per decidir la satisfactibilitat de fórmules Booleanes en CNF mitjançant el procediment de decisió DPLL.

Les variables Booleanes es representaran en el nostre programa Prolog amb nombres positius. Els literals seran nombres enters, les clàusules seran llistes de literals i les CNF llistes de clàusules. Per exemple, la fórmula CNF: $(P \vee Q \vee \neg R) \wedge (\neg Q \vee R)$ la representarem amb prolog com a `[[1,2,-3],[-2,3]]`. Adoneu-vos que la clàusula buida seria una llista buida. Una interpretació la representarem amb Prolog com a una llista de literals, per exemple, per a la fórmula anterior, la interpretació $P = 1, Q = 0, R = 1$ la representarem com a `[1,-2,3]`. Fixeu-vos també que aquesta interpretació seria un model de la CNF mentre que aquesta altra: `[-1,2,-3]` no en seria model.

Completeu el predicat `sat(F, I, M)` que donada una fórmula `F` en CNF i una interpretació `I`, es satisfaci quan `M` sigui un model de `F` construit extenent `I` (la primera crida serà de la forma `sat(CNF, [ ], M)`).

Aquest predicat implementa un SAT solver mitjançant el procediment DPLL. Aquest procediment va construint un model i eliminant de la CNF les clàusules ja satisfetes mentre en quedin i no s'hagi trobat la clàusula buida. Quan s'ha triat un literal per posar al model, s'ha de simplificar la CNF tenint en compte el literal triat:

- les clàusules que tenen aquell literal ja es poden eliminar de la CNF (fixeu-vos que ja estaran satisfetes perquè el literal que hem triat les satisfà)

- a les clàusules on apareix el literal que hem triat però negat, els hem de treure aquest literal ja que a la interpretació hi apareix negat. És a dir, la clàusula seguirà a la CNF però amb un literal menys. De fet és en aquest moment on pot apareixer la clàusula buida.

Per exemple, suposem que tenim la CNF `[[1],[2,-3, 4],[-2,3],[4,5],[-2,-3]]`. **Sempre que es pugui s'ha de triar un literal que aparegui sol**, és a dir, a una clàusula unitària, **sino es pot, un altre qualsevol**, per exemple el primer de la primera clàusula o aquest mateix negat. Per tant aquí triaríem el literal `1` i simplificaríem la CNF que quedaria: `[[2,-3, 4],[-2,3],[4,5],[-2,-3]]`. Llavors podríem triar, per exemple, el literal `2` per afegir a la interpretació que anem construint. La nova CNF després de simplificar seria: `[[3],[4,5],[-3]]`. Ara DPLL triaria `3` (o bé `-3` que també apareix sol, però a l'exemple fem que triem d'esquerra a dreta i per tant triem `3`). Després de simplificar la CNF li quedaria la fórmula: `[[4,5],[ ]]`, per tant **apareixeria la**

clàusula buida i hauríem de fer backtraking (és a dir heu de fer que el programa Prolog faci backtraking) per mirar si pot triar algun altre literal en algun altre moment. El que hauria de passar és que pogués fer backtraking fins a triar `-2` obtenint després de simplificar: `[[-3, 4],[4,5]]` (fixeu-vos que no te sentit fer backtraking al haver triat un literal d'una clàusula unitària, per això el backtraking es fa a la tria del literal `2`). Llavors triaria `-3` i quedaria la CNF `[[4,5]]`, i finalment triaria `4` quedant `[]`, és a dir la CNF trivialment satisfactible. El model obtingut seria: `[1,-2,-3,4]`. (Vegeu els apunts sobre DPLL per a més detalls).

El codi a completar és el següent:

```
sat([],I,I):- write("SAT") ,nl,!.

sat(CNF,I,M):-
    % Ha de triar un literal d'una clausula unitaria, si no n'hi ha cap,
    % llavors un literal pendent qualsevol.

    tria(CNF,Lit),

    % Simplifica la CNF amb el Lit triat (compte pq pot fallar, es a dir
    % si troba la clausula buida fallara i fara backtraking).
    % Aquesta simplificacio fara el seguent:
    %  - eliminara les clausules que tinguin el literal
    %  - eliminara el literal negat de totes les clausules, es aqui on
    %     pot sortir la clausula buida i per tant hauria de fallar i fer
    %     backtraking

    simplif(Lit,CNF,CNFS),

    % crida recursiva amb la CNF i la interpretacio actualitzada

    sat( ... , ..., M ).
```

k-colorabilitat

Per a codificar el problema de colorejar un graf amb k colors en CNF el que farem serà utilitzar variables Booleanes que representaran quin color assignem a cada node. Per exemple, la variable Booleana $X_{i,j}$ ens servirà per codificar que el node i el coloregem amb el color j. D'aquesta manera per a cada node i tindrem les k variables $X_{i,1}, X_{i,2}, \ldots X_{i,k}$ per a poder representar quin color li hem assignat. En el nostre cas, com ja hem dit al descriure el SAT solver, les variables seran nombres enters, per tant en realitat per representar les variables $X_{i,1}, X_{i,2}, \ldots X_{i,k}$, utilitzarem, per exemple, `1,2,3...k` per a les variables del primer node, `k+1,k+2,...2·k` per a les del segon node, etc.

La nostra CNF haurà de forçar que **a cada node se li hagi assignat exactament un color**. Per tant ens caldrà fer el predicat: **unCert(+Xs,F)** de manera que donada una llista de variables Booleanes `Xs`, `F` sigui la fórmula CNF que es satisfaci quan (és a dir, que faci que) exactament una de les variables d'`Xs` sigui certa i totes les altres falses.

Per exemple, `unCert([1,2,3],F)` hauria de construir una CNF `F` que tingués com a model `[1,-2,-3]` o `[-1,2,-3]` o `[-1,-2,3]`.

Haurem de considerar que potser hi ha alguns examens amb dies fixats, o dit d'una altra manera, alguns **nodes amb colors ja inicialitzats**. Haureu de fer el predicat **inicialitza(+LLV,+Ini,CNF)** de manera que donada una llista de llistes de variables (les llistes d'$X_{i,j}$ que haurem creat abans), i una llista de parelles (nombre de node, color), generi una CNF que faci que cada node inicialitzat tingui el color que es demana (els colors també seran nombres).

Un cop fet això ens caldrà forçar que **els nodes veins tinguin colors diferents**. Haureu de fer el predicat **fesMutexes(+LLV,+Arestes,CNF)** de manera que donada la llista de llistes de variables, i una llista d'arestes (les arestes seran parelles de nombres de nodes), en generi una CNF que eviti que dos veins tinguin el mateix color assignat.

Ara ja podrieu fer el predicat **codifica(N,K,Arestes,Inici,CNF)** que donat un nombre de nodes `N`, un nombre de colors permesos `K`, una llista d'`Arestes` (parelles de nombres d'1 a `N`), i una llista de parelles `Inici` (nombre de node, color), ens generi la CNF que es satisfactible si i només si el graf format per les `Arestes`, i amb alguns nodes inicialitzats segons `Inici`, es pot colorejar amb `K` colors.

Nombre Cromàtic

Feu el predicat **resol(N,Arestes,Inici)** que utilitzi el predicat `codifica` per a descobrir el nombre cromàtic del graf donat i mostrar la solució que hagi trobat. És a dir, volem saber per cada color quins nodes te (això ens permetrà saber quins examens es faran cada dia). En l'exemple del graf que donem, la sortida (els nodes estan numerats en sentit horari essent el de dalt de tot l'1 i el de sota seu el 6)hauria de ser quelcom com ara:

```
?- resol(10,[(1,2),(2,3),(3,4),(4,5),(5,1),(1,6),(2,7),(3,8),(4,9),(5,10),
             (6,8)(7,9),(8,10),(9,6),(10,7)],[]).

El nombre chromatic és: 3
color 1: 1, 4, 7, 8
color 2: 2, 5, 6
color 3: 3, 9, 10

true ?;

color 1: ...
```

com veieu el predicat ha de ser *backtrakejable* i per tant ha de donar totes les coloracions possibles.

A part dels predicats demanats, possiblement caldrà que en feu d'altres d'auxiliars. **Feu servir com a base del programa el fitxer** `chromatic.pl` que conté comentaris i suggerències més pas a pas de com completar la pràctica.

Us recomano que verifiqueu individualment els predicats que aneu fent per estar segurs que funcionen, el *debugging* amb Prolog pot ser dur amb programes grans ...

Es valorarà positivament la programació "elegant" i eficient.

Cal un ús adequat dels talls per tal de ser eficients però sense deixar solucions fora.

Lliurament

- Les pràctiques **S'HAN** de fer en equips de dos.
- Cal que documenteu el codi.
- S'haurà de lliurar un document amb el codi comentat i les execucions que es demanen mostrant els resultats obtinguts (captures de pantalla). En concret **el document caldrà que tingui els següents apartats**:
  1. Descripció del problema resolt i abast de la vostra solució: breument definir el problema que heu resolt, incloent les possibles extensions o limitacions de la vostra solució.
  2. Codi comentat: codi breument comentat, ben indentat i llegible.
  3. Particularitats del codi: descripció detallada dels predicats més rellevants.
  4. Jocs de prova: descripció dels jocs de prova i "pantallassos" d'us. Es valorarà algun joc de prova amb nombre cromàtic alt.
  5. Referències de bibliografia i/o llibreries usades si s'escau.
- **Les entregues molt probablement seran presencials** per poder avaluar la implicació en la pràctica per part dels membres de l'equip. També es possible que per acabar d'avaluar se us faci fer alguna modificació de la pràctica *in situ*.
- Data de lliurament (pel moodle): **1 de juny.**