Exercise session 11: Grounding and CDCL

Ingmar Dasseville *ingmar.dasseville@cs.kuleuven.be*
Thomas Vanstrydonck *thomas.vanstrydonck@cs.kuleuven.be*
Simon Marynissen *simon.marynissen@cs.kuleuven.be*

December 19, 2017

1 Exercise 1: Grounding

The following theory describes the graph coloring problem. On Toledo you can find an explanation of the rewriting schemes used to transform FO(.) to (E)CNF[1].

- Normalize the theory, you don't need to do introduce tseitins
- Ground this theory for the given structure S
- Compare your grounding to the one actually generated by IDP. You can see IDP's grounding by executing the following main program:
  `stdoptions.cpsupport = false`
  `printgrounding(T,S)`
  Is your grounding smaller or larger?
- Use DPLL to find all models for this structure

```
vocabulary V {
    type Color isa string
    type Area isa int
    Border ( Area , Area )
    Coloring ( Area , Color )
}
theory T : V {
    ? x : Coloring (2 ," B ") | Coloring (1 , x ).
    ! area :
      (? col : Coloring ( area , col ) &
      (~? x : ? c : ? c2 :
          ( c ~= c2 & Coloring (x , c2 )) & Coloring (x , c ) )).
    ~(? x1 x2 c : Border ( x1 , x2 ) & Coloring ( x1 , c ) & Coloring ( x2 , c )).
}
structure S : V {
    Area ={1;2;3}
    Color ={ R ; G ; B }
    Border ={
        (1 ,2);
        (2 ,3);
        (1 ,3)
    }
    Coloring < cf > = {1 , B }
```

[1] IDP supports more than pure CNF to better support definitions, you don't need these features in this exercise

```
    Coloring < ct > = {2 , B }
}
```

2 Exercise 2: Tseitin transformation

In exercise two, we transformed a theory into CNF by the standard algorithm. In some cases, using tseitin transformation might be a better choice. Consider for example the following theory T:

$$\begin{aligned} (a \wedge b) \vee c &\Leftrightarrow a \\ d \vee e &\Rightarrow (a \wedge b) \vee c \end{aligned}$$

If we introduce a new (*tseitin*) variable t to replace $(a \wedge b) \vee c$, the theory (T') becomes:

$$\begin{aligned} t &\Leftrightarrow a \\ d \vee e &\Rightarrow t \\ t &\Leftrightarrow (a \wedge b) \vee c \end{aligned}$$

Given a partial interpretation J, executing unit propagation on T' instead of on T can lead to more propagations, even if t is unknown in J! Can you find such a J?
What is the essential property that leads to this stronger propagation? Hint: consider how you would apply tseitin transformation to the slightly different theory T'':

$$\begin{aligned} (a \wedge b) \vee c &\Leftrightarrow a \\ d \vee e &\Rightarrow \neg c \wedge (\neg a \vee \neg b) \end{aligned}$$

Are the theories T and T' logically equivalent? If this is the case, prove it. If not, what weaker forms of "equivalence" exist between the theories?

3 Exercise 3: Clause Learning

Given the following CNF formula, complete the table and identify the conflict clause. Here *Time* represents the time at which the literal's value is set. The level of a literal is the number of decision literals at that time, starting from 0. All unit literals have the same level as the decision literal from which they were derived. In the Table, the literal A is set to *false* at time 3, it is the second decision literal.

$$\begin{aligned} &\neg F \vee G \\ &A \vee B \vee C \\ &D \vee \neg F \vee E \\ &A \vee B \vee \neg C \end{aligned}$$

	Value	Time	Level
A	F	3	2
B	F	6	4
C	...	...	...
D	F	4	3
E	...	...	...
F	T	1	1
G	...	...	...

- Conflict clause:
- Learned clause:
- What level does the algorithm backjump to?

After adding the learned clause and backjumping to the appropriate level the algorithm continues, the time is not reset. This results in the following table.

	Value	Time	Level
A	F	3	2
B	T	8	2
C	...	...	...
D	...	...	...
E	...	...	...
F	T	1	1
G	T	2	1

4 Exercise 4: Extended DPLL

Given the CNF formula:

$$
\begin{aligned}
&a \vee b \\
&\neg b \vee c \vee d \\
&\neg b \vee e \\
&\neg d \vee \neg e \vee f \\
&\neg a \vee g \\
&\neg g \vee b \\
&\neg h \vee j
\end{aligned}
$$

Apply the DPLL + clause learning + nonchronological backtracking algorithm to find models of this theory. To start the algorithm, sequentially choose c, f, h, a to be false (in that order). Afterwards, continue applying the algorithm until you have learned at least two clauses.

A variation on the algorithm continues resolution until the decision literal itself is the only literal in the learned clause that is on the current decisionlevel. In practice, this algorithm takes more time to find models. Apply this adapted version on the same CNF formula and find out why.