VU Formale Methoden der Informatik

Block 2: Satisfiability Problems

Bernhard Urban

Matr.Nr.: 0725771

lewurm@gmail.com

May 21, 2011

Exercise 1: *Entailment, Equivalence*

For the formulas on the left, mark with Yes or No in the table on the right whether the indicated logical releations hold:

(1) $\forall x P(x) \vee \forall x Q(x,x)$

(2) $\forall x(P(x) \vee Q(x,x))$

(3) $\forall x(\forall z P(z) \wedge \forall y Q(x,y))$

(4) $\exists y \forall x P(x,y)$

(5) $\forall x \exists y P(x,y)$

	Yes	No
(1) $\Rightarrow$ (2)	X	
(2) $\Rightarrow$ (3)		X
(3) $\Rightarrow$ (1)	X	
(4) $\Rightarrow$ (5)	X	
(5) $\Rightarrow$ (4)		X

Analogously mark in the following table whether the equivalences hold:

	Yes	No
$\forall x \forall y F \Leftrightarrow \forall y \forall x F$	X	
$\forall x \exists y F \Leftrightarrow \exists x \forall y F$		X
$\exists x \exists y F \Leftrightarrow \exists y \exists x F$	X	
$\forall x F \vee \forall x G \Leftrightarrow \forall x(F \vee G)$		X
$\forall x F \wedge \forall x G \Leftrightarrow \forall x(F \wedge G)$	X	
$\exists x F \vee \exists x G \Leftrightarrow \exists x(F \vee G)$	X	
$\exists x F \wedge \exists x G \Leftrightarrow \exists x(F \wedge G)$		X

Exercise 2: *Formulas and Structures*

(a) We define a predicate $M(x)$ if x is an element of M:

$$I(M(x)) = \ x \in M$$

Therefore we can write $\exists x \in M : P(x)$ in first-order logic as follows:

$$\exists x(M(x) \land P(x))$$

and $\forall x \in M : P(x)$ as:

$$\forall x(M(x) \to P(x)) = \forall x(\neg M(x) \lor P(x))$$

Also we should show that the following equivalence hold for our translation:

$$\begin{aligned} \neg\exists x \in M : P(x) &\iff \forall x \in M : \neg P(x) \\ \neg\exists(M(x) \land P(x)) &\iff \forall x(M(x) \to \neg P(x)) \\ \forall x \neg(M(x) \land P(x)) &\iff \forall x(\neg M(x) \lor \neg P(x)) \\ \forall x(\neg M(x) \lor \neg P(x)) &\iff \forall x(\neg M(x) \lor \neg P(x)) \end{aligned}$$

(b) Present models for the following formulas:

(i) $\neg\exists x \forall y P(x, y)$
$\mathcal{U} = \mathbb{N}_0$ $\quad I(P(x, y)) = \ x > y$

(ii) $\exists x(Q(x, c) \land \neg\forall y Q(x, y))$
$\mathcal{U} = \mathbb{N}_0$ $\quad I(P(x, y)) = \ x < y$ $\quad I(c) = 1$

(iii) $\forall x \forall y(P(x, y) \to P(x, f(y))) \land \forall x \exists y P(x, y)$
$\mathcal{U} = \mathbb{N}_0$ $\quad I(P(x, y)) = \ x < y$ $\quad I(f(x)) = x$

(c) Find a formula F in first-order logic such that the universe of any interpretation that is a model for F has:

(i) ... *at least* two elements.

$$F_i = \exists x \exists y(P(x) \land \neg P(y))$$

(ii) ... *at least* n elements.

$$F_{ii} = \exists x_1 \exists x_2 \ldots \exists x_n \bigwedge_{i=0}^{n} \bigwedge_{j=0,\, j \neq i}^{n} P_i(x_i) \land \neg P_j(x_i)$$

(iii) ... infinitely many elements.

$$F_{iii} = \underbrace{\forall x \neg P(x, x)}_{\text{irreflexivity}} \land \underbrace{\forall x \forall y \forall z(P(x, y) \land P(y, z) \to P(x, z))}_{\text{transitivity}} \land \forall x \exists y P(x, y)$$

Exercise 3: *Modelling and Solving*

(a) The following formula F' describes all puzzles of size 9×9 are described:

$$F' = F_{\text{Row}} \wedge F_{\text{Column}} \wedge F_{\text{Block}} \wedge F_{\text{Field}}$$

$$F_{\text{Row}} = \bigwedge_{x=1}^{9} \bigwedge_{z=1}^{9} \bigvee_{y=1}^{9} v_{x,y,z}$$

$$F_{\text{Column}} = \bigwedge_{y=1}^{9} \bigwedge_{z=1}^{9} \bigvee_{x=1}^{9} v_{x,y,z}$$

$$F_{\text{Block}} = \bigwedge_{lx=0}^{2} \bigwedge_{ly=0}^{2} \bigwedge_{z=1}^{9} \bigvee_{x=1}^{3} \bigvee_{y=1}^{3} v_{(lx\cdot 3)+x,(ly\cdot 3)+y,z}$$

$$F_{\text{Field}} = \bigwedge_{x=1}^{9} \bigwedge_{y=1}^{9} \bigwedge_{z=1}^{8} \bigwedge_{i=z+1}^{9} (\neg v_{x,y,z} \vee \neg v_{x,y,i})$$

(b) In order to solve the given puzzle, I wrote a small program which generates all clauses of F and G (given in the assigmnet) in the DIMACS format.

Each cell is encoded with its number ($v_{x,y,z}$) in the following way:

$$c_{x,y,z} = x \cdot 10 \cdot 10 + y \cdot 10 + (z+1) \quad \text{where } x, y, z \in \{0, 1, \ldots, 8\}$$

Thus, we can easily extract all “positive” literals of the `minisat` output, however, this method leaves some gaps: We can “elimate” them by the following clauses:

$$F_{\text{Fill}} = \bigwedge_{x=1}^{9} \bigwedge_{y=1}^{9} \neg c_{x,y,0}$$

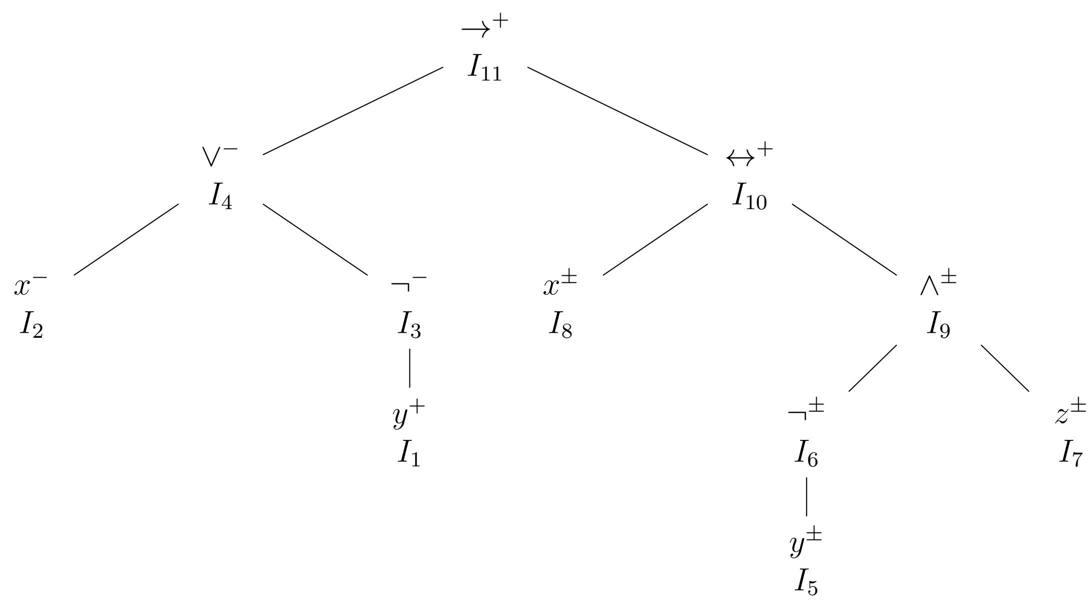
For the sourcecode, Makefile and a sample output see the Appendix.

The solution for the given puzzle is:

6	8	3	1	9	7	5	4	2
1	4	5	6	3	2	9	8	7
7	2	9	8	5	4	6	1	3
8	9	6	7	4	3	2	5	1
5	3	7	2	6	1	4	9	8
4	1	2	5	8	9	3	7	6
3	7	4	9	2	8	1	6	5
2	6	1	4	7	5	8	3	9
9	5	8	3	1	6	7	2	4

Exercise 4: *CNF Transformation*

(1) Label the formula tree:



(2) Translate the "labeling formulas" to clauses:

Equivalences for SFOs[1] in ϕ	Associated Clauses $C_1(\phi)$	$C_2(\phi)$	$C_3(\phi)$	$C_4(\phi)$
$I_1 \leftrightarrow y$	$\neg I_1 \vee y$	$I_1 \vee \neg y$		
$I_2 \leftrightarrow x$	$\neg I_2 \vee x$	$I_2 \vee \neg x$		
$I_3 \leftrightarrow \neg I_1$	$\neg I_3 \vee \neg I_1$	$I_3 \vee I_1$		
$I_4 \leftrightarrow I_2 \vee I_3$	$\neg I_4 \vee I_2 \vee I_3$	$I_4 \vee \neg I_2$	$I_4 \vee \neg I_3$	
$I_5 \leftrightarrow y$	$\neg I_5 \vee y$	$I_5 \vee \neg y$		
$I_6 \leftrightarrow \neg I_5$	$\neg I_6 \vee \neg I_5$	$I_6 \vee I_5$		
$I_7 \leftrightarrow z$	$\neg I_7 \vee z$	$I_7 \vee \neg z$		
$I_8 \leftrightarrow x$	$\neg I_8 \vee x$	$I_8 \vee \neg x$		
$I_9 \leftrightarrow I_6 \wedge I_7$	$\neg I_9 \vee I_6$	$\neg I_9 \vee I_7$	$I_9 \vee \neg I_6 \vee \neg I_7$	
$I_{10} \leftrightarrow I_8 \leftrightarrow I_9$	$\neg I_{10} \vee \neg I_8 \vee I_9$	$\neg I_{10} \vee I_8 \vee \neg I_9$	$I_{10} \vee \neg I_8 \vee \neg I_9$	$I_{10} \vee I_8 \vee I_9$
$I_{11} \leftrightarrow I_4 \to I_{10}$	$\neg I_{11} \vee \neg I_4 \vee I_3$	$I_{11} \vee I_4$	$I_{11} \vee \neg I_{10}$	

(3) **(a)** Tseitin: All clauses in the table above get conjuncted.

[1] subformula occurence

(b) Plaisted and Greenbaum:

$$\begin{aligned}
&(\neg I_1 \lor y) \land \\
&(I_2 \lor \neg x) \land \\
&(I_3 \lor I_1) \land \\
(I_4 \lor \neg I_2) \land &(I_4 \lor \neg I_3) \land \\
(\neg I_5 \lor y) \land &(I_5 \lor \neg y) \land \\
(\neg I_6 \lor \neg I_5) \land &(I_6 \lor I_5) \land \\
(\neg I_7 \lor z) \land &(I_7 \lor \neg z) \land \\
(\neg I_8 \lor x) \land &(I_8 \lor \neg x) \land \\
(\neg I_9 \lor I_6) \land (\neg I_9 \lor I_7) \land &(I_9 \lor \neg I_6 \lor \neg I_7) \land \\
(\neg I_{10} \lor \neg I_8 \lor I_9) \land &(\neg I_{10} \lor I_8 \lor \neg I_9) \land \\
&(\neg I_{11} \lor \neg I_4 \lor I_3)
\end{aligned}$$

Exercise 5: *DPLL Procdures*

(a) F_1

$$F_1 = \underbrace{(A \lor \neg B)}_{c_1} \land \underbrace{(C \lor \neg A)}_{c_2} \land \underbrace{(A \lor B)}_{c_3} \land \underbrace{(\neg A \lor \neg C)}_{c_4}$$

Since $C_{Ap} \not> C_{An}$ we choose $A = \textsf{false}$ according to the DLIS heuristic, we have to set $A = 0@1$. Because of $C_{Bp} \not> C_{Bn}$ we choose $B = \textsf{false}$ for the next step (cf. c_1), however this decision conflicts with c_3, thus we have to go back to the first UIP[2] $A = 0@1$ and learn that $A = 1@1$.
Therefore, we set $C = \textsf{false}$ (since $C_{Cp} \not> C_{Bn}$) and run into a similiar conflict as before. Eventually, F_1 is unsatisfiable.

(b) F_2

$$F_2 = \underbrace{(A \lor \neg E)}_{c_1} \land \underbrace{(\neg C \lor \neg D)}_{c_2} \land \underbrace{(\neg A \lor B \lor D)}_{c_3} \land \underbrace{(C \lor D \lor \neg E)}_{c_4} \land \underbrace{(\neg A \lor E)}_{c_5} \land \underbrace{(\neg B \lor \neg E)}_{c_6}$$

Since $C_{Dp} \not> C_{En}$ we choose $E = \textsf{false} \to E = 0@1$. Due to the unit clause c_5 we have to say that $A = 0@1$. Thus, just the clause c_2 is left over. We can choose between C and D; according to the assignment we should decide by lexical ordering $\to C = 0@2$. The remaining variables can have any value, but we should choose $\textsf{true}$: $B = 1@2$, $D = 1@2$. Finally we know that F_2 is satisfiable.

[2]unique implication point

Exercise 6: *UIP*

Assume that there are two unique implication points (UIP) for an decision node $\mathcal{D}$ and a conflict node $\mathcal{C}$ in a implication graph $G(V, E)$, i.e. $u_1, u_2, \mathcal{D}, \mathcal{C} \in V$. Therefore, all paths from $\mathcal{D}$ to $\mathcal{C}$ must contain u_1 and u_2 too. We also assume that both UIPs have the same distance to $\mathcal{C}$. From that we can follow, that all paths from $\mathcal{D}$ to $\mathcal{C}$ must visit u_1 and u_2 which means in G a cycle exists. However, we know that G is a directed acyclic graph by definition and thus doesn't contain any cycles $\Rightarrow u_1 = u_2$.

Exercise 7: *Ackermann's and Bryant's Reductions*

Reduce the problem of validity of φ^{UF} to the problem of validity in equality logic.

Ackermann's Reductions

$$\begin{gathered}\varphi^{UF} = F(F(x_1)) \neq F(x_1) \wedge F(F(x_1)) \neq F(x_2) \wedge x_2 = F(x_1) \wedge \\ G(x_1, F(x_2)) = F(G(x_1, x_2)) \wedge G(x_1, x_1) = F(x_1)\end{gathered}$$

First we number the instances of the UF:

$$\begin{gathered}\varphi^{UF} = F_2(F_1(x_1)) \neq F_1(x_1) \wedge F_2(F_1(x_1)) \neq F_3(x_2) \wedge x_2 = F_1(x_1) \wedge \\ G_1(x_1, F_3(x_2)) = F_4(G_2(x_1, x_2)) \wedge G_3(x_1, x_1) = F_1(x_1)\end{gathered}$$

Compute $flat^E$ by replacing UF F_i by the new variable f_i:

$$\begin{gathered}flat^E = f_2 \neq f_1 \wedge f_2 \neq f_3 \wedge x_2 = f_1 \wedge \\ g_1 = f_4 \wedge g_3 = f_1\end{gathered}$$

Add functionality constraints, i.e., compute FC^E:

$$\begin{aligned}
(x_1 = x_1 \wedge f_3 = x_2 \rightarrow g_1 = g_2) &\qquad \wedge \\
(x_1 = x_1 \wedge f_3 = x_1 \rightarrow g_1 = g_3) &\qquad \wedge \\
(x_1 = x_1 \wedge x_2 = x_1 \rightarrow g_2 = g_3) &\qquad \wedge \\
(x_1 = f_1 \rightarrow f_1 = f_2) &\qquad \wedge \\
(x_1 = x_2 \rightarrow f_1 = f_3) &\qquad \wedge \\
(x_1 = g_2 \rightarrow f_1 = f_4) &\qquad \wedge \\
(f_1 = x_2 \rightarrow f_2 = f_3) &\qquad \wedge \\
(f_1 = g_2 \rightarrow f_2 = f_4) &\qquad \wedge \\
(x_2 = g_2 \rightarrow f_3 = f_4) &
\end{aligned}$$

Now we have $\varphi^E : \; FC^E \rightarrow flat^E$

Bryant's Reductions

For the numbering part see above. Compute $flat^E$ by replacing UF F_i by the new term variable F_i^*:

$$\begin{aligned} flat^E = & F_2^* \neq F_1^* \wedge F_2^* \neq F_3^* \wedge x_2 = F_1^* \wedge \\ & G_1^* = F_4^* \wedge G_3^* = F_1^* \end{aligned}$$

Compute F_i^* and G_j^*:

$$F_1^* = f_1 \qquad F_2^* = \begin{pmatrix} \text{case} & x_1 = F_1^* & : f_1 \\ & \mathsf{true} & : f_2 \end{pmatrix}$$

$$F_3^* = \begin{pmatrix} \text{case} & x_1 = x_3 & : f_1 \\ & F_1^* = x_3 & : f_2 \\ & \mathsf{true} & : f_3 \end{pmatrix} \qquad F_4^* = \begin{pmatrix} \text{case} & x_1 = G_2^* & : f_1 \\ & F_1^* = G_2^* & : f_2 \\ & x_2 = G_2^* & : f_3 \\ & \mathsf{true} & : f_4 \end{pmatrix}$$

$$G_1^* = g_1 \qquad G_2^* = \begin{pmatrix} \text{case} & x_1 = x_1 \wedge F_3^* = x_2 & : g_1 \\ & \mathsf{true} & : g_2 \end{pmatrix}$$

$$G_3^* = \begin{pmatrix} \text{case} & x_1 = x_1 \wedge F_3^* = x_1 & : g_1 \\ & x_1 = x_1 \wedge x_2 = x_1 & : g_2 \\ & \mathsf{true} & : g_3 \end{pmatrix}$$

Then $\varphi^E : \ (\bigwedge_{i=1}^4 F_i^* \ \wedge \ \bigwedge_{j=1}^3 G_j^*) \rightarrow flat^E$

Exercise 8: *Sparse Method*

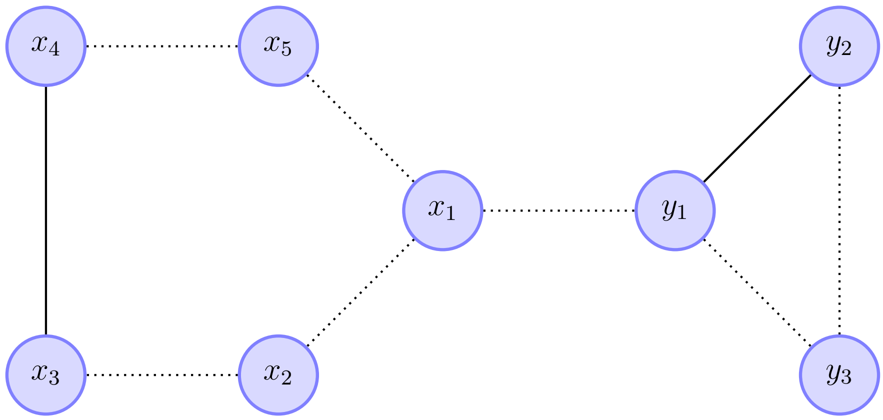
For the following formula in equality logic φ^E we should compute an equisatisfiable formula in propositional logic:

$$\begin{aligned} \varphi^E : & (x_1 = x_2 \vee x_2 = x_3) \wedge (x_3 \neq x_4 \vee x_4 = x_5 \vee x_5 = x_1) \wedge \\ & (x_1 = y_1 \vee y_1 \neq y_2) \wedge (y_1 = y_3 \vee y_2 = y_3) \end{aligned}$$

From that, we can compute $E_=$ (equality literals) and $E_{\neq}$ (disequality literals) of φ^E easily:

$$\begin{aligned} & E_= = \{x_1 = x_2,\ x_2 = x_3,\ x_4 = x_5,\ x_5 = x_1,\ x_1 = y_1,\ y_1 = y_3,\ y_2 = y_3\} \\ & E_{\neq} = \{x_3 \neq x_4,\ y_1 \neq y_2\} \end{aligned}$$

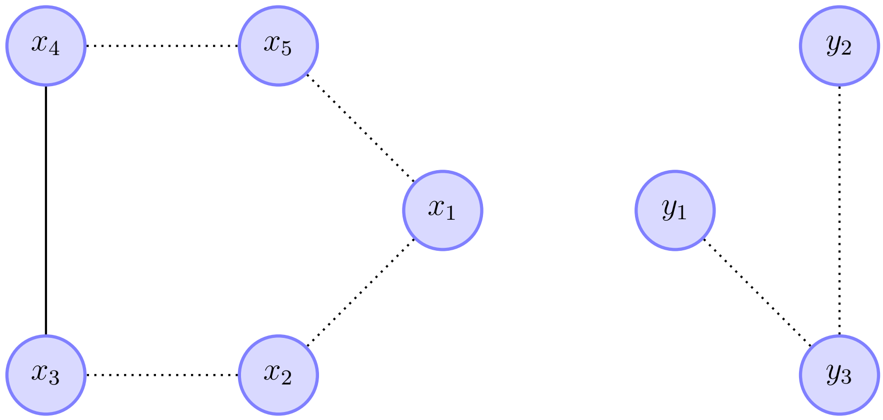
At this point we can construct an equality graph $G^E(\varphi^E) = (V, E_=, E_{\neq})$:



There are two contradictory cycles[3] in the graph. We can simplify the graph, by setting the literals of the corresponding edges which are not part of any contradictory cycle to true. In our case the edge (x_1, y_1) is such one:

$$\begin{aligned}
\varphi_1^E &: (x_1 = x_2 \lor x_2 = x_3) \land (x_3 \neq x_4 \lor x_4 = x_5 \lor x_5 = x_1) \land \\
&\quad (\mathsf{true} \lor y_1 \neq y_2) \land (y_1 = y_3 \lor y_2 = y_3) \\
\varphi_1^E &: (x_1 = x_2 \lor x_2 = x_3) \land (x_3 \neq x_4 \lor x_4 = x_5 \lor x_5 = x_1) \land \\
&\quad (y_1 = y_3 \lor y_2 = y_3)
\end{aligned}$$

Therefore, $G^E(\varphi_1^E)$ looks like:

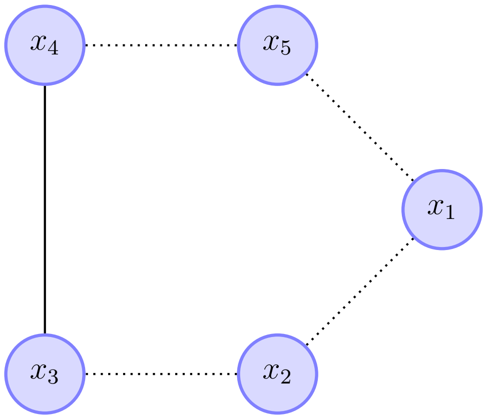


[3] a cycle with exactly one disequality edge

As we can see, two more edges can be removed, namely (y_1, y_3) and (y_3, y_2):

$$
\begin{aligned}
\varphi_2^E &: (x_1 = x_2 \vee x_2 = x_3) \wedge (x_3 \neq x_4 \vee x_4 = x_5 \vee x_5 = x_1) \wedge \\
&\quad (\mathsf{true} \vee \mathsf{true}) \\
\varphi_2^E &: (x_1 = x_2 \vee x_2 = x_3) \wedge (x_3 \neq x_4 \vee x_4 = x_5 \vee x_5 = x_1)
\end{aligned}
$$

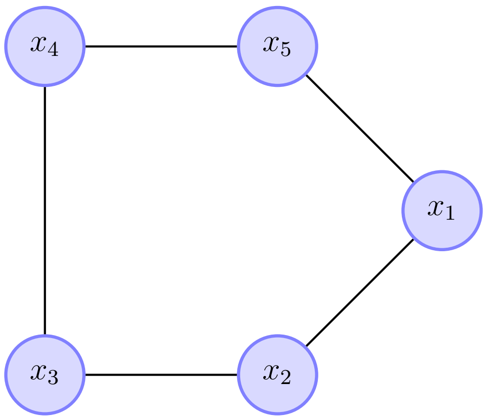
which results in the following graph:



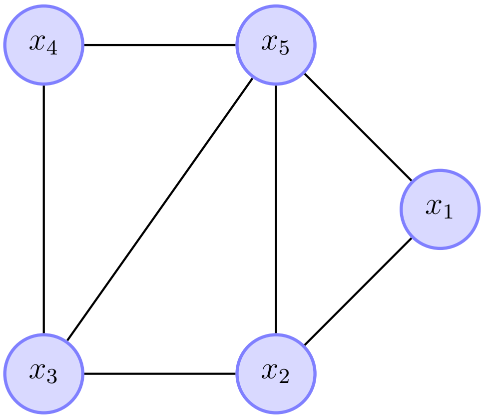
After we have simplified the graph we have to construct a boolean formula $e(\varphi_2^E)$ by replacing $x_i = x_j$ in φ_2^E by a boolean variable $e_{i,j}$:

$$e(\varphi_2^E) = (e_{1,2} \vee e_{2,3}) \wedge (\neg e_{3,4} \vee e_{4,5} \vee e_{5,1})$$

Next we construct the nonpolor equality graph $G_{NP}^E(\varphi_2^E)$:



Now we make $G^E_{NP}(\varphi^E_2)$ chordal:



In the final steps we do the following:

- Set B_t to `true`
- For each triangle $(e_{i,j}, e_{j,k}, e_{k,i})$ in $G^E_{NP}(\varphi^E_2)$ do

$$\begin{aligned} B_t := {} & (e_{i,j} \wedge e_{j,k} \rightarrow e_{i,k}) \wedge \\ & (e_{i,j} \wedge e_{i,k} \rightarrow e_{j,k}) \wedge \\ & (e_{i,k} \wedge e_{j,k} \rightarrow e_{i,j}) \wedge B_t \end{aligned}$$

- Return $e(\varphi^E_2) \wedge B_t$

Therefore we get for $G^E_{NP}(\varphi^E_2)$:

$$\begin{aligned} B_t &= (e_{1,2} \wedge e_{5,1} \rightarrow e_{2,5}) \wedge (e_{5,1} \wedge e_{2,5} \rightarrow e_{1,2}) \wedge (e_{2,5} \wedge e_{1,2} \rightarrow e_{5,1}) \\ &= (e_{2,3} \wedge e_{5,2} \rightarrow e_{3,5}) \wedge (e_{5,2} \wedge e_{3,5} \rightarrow e_{2,3}) \wedge (e_{3,5} \wedge e_{2,3} \rightarrow e_{5,2}) \\ &= (e_{3,4} \wedge e_{5,3} \rightarrow e_{4,5}) \wedge (e_{5,3} \wedge e_{4,5} \rightarrow e_{3,4}) \wedge (e_{4,5} \wedge e_{3,4} \rightarrow e_{5,3}) \end{aligned}$$

Finally, the following holds:

$$\varphi^E \text{ is satisfiable iff } e(\varphi^E_2) \wedge B_t$$

Sourcecode of `sdk.lhs`

```
import Data.List
import Text.Printf

sdksize :: Int
sdksize = 9

type UnLit = (Int, Int, Int)
```

```haskell
type Lit = Int
type Clause = [Lit]

main :: IO ()
main = do
  printf "c sup? some sudoku tonight, eh? lulz\n"
  let solution = (genRow ⧺ genCol ⧺ genBlock ⧺ genField ⧺ genStart ⧺ genFill)
  let solstr = concat $ map showClause solution
  printf "p cnf 999 %d\n" (length solution)
  printf "%s" solstr

showClause :: Clause → String
showClause (x : xs)
   | x ≡ 0 = "0\n"
   | otherwise = (show x) ⧺ " " ⧺ (showClause xs)

encode :: UnLit → Lit
encode (x, y, z) = (x ∗ 10 ∗ 10) + (y ∗ 10) + z

genRow :: [Clause]   -- cf. F_Row
genRow =
  [[encode (x, y, z) | y ← [1 . . sdksize]] ⧺ [0]
   | x ← [1 . . sdksize], z ← [1 . . sdksize]]

genCol :: [Clause]   -- cf. F_Column
genCol =
  [[encode (x, y, z) | x ← [1 . . sdksize]] ⧺ [0]
   | y ← [1 . . sdksize], z ← [1 . . sdksize]]

genBlock :: [Clause]   -- cf. F_Block
genBlock =
  [[encode ((lx ∗ 3) + x, (ly ∗ 3) + y, z) | y ← [1 . . sub], x ← [1 . . sub]] ⧺ [0]
   | lx ← [0 . . (sub − 1)], ly ← [0 . . (sub − 1)], z ← [1 . . sdksize]]
  where sub = sdksize `div` 3

genField :: [Clause]   -- cf. F_Field
genField =
  [(−encode (x, y, z)) : (−encode (x, y, i)) : [0]
   | x ← [1 . . sdksize], y ← [1 . . sdksize], z ← [1 . . (sdksize − 1)], i ← [(z + 1) . . sdksize]]

genStart :: [Clause]
genStart = map (flip (:) [0] ∘ encode) [
  (1, 1, 6), (1, 8, 4), (2, 3, 5), (2, 6, 2), (2, 9, 7), (3, 1, 7), (3, 2, 2),
  (3, 3, 9), (3, 9, 3), (4, 2, 9), (4, 5, 4), (4, 9, 1), (5, 5, 6), (6, 1, 4),
  (6, 5, 8), (6, 8, 7), (7, 1, 3), (7, 7, 1), (7, 8, 6), (7, 9, 5), (8, 1, 2),
  (8, 4, 4), (8, 7, 8), (9, 2, 5), (9, 9, 4)]

  -- for better readability of the output we use natural numbering of the
  -- literals, however we have to “fill” the gaps and say they shouldn’t
  -- be true ever.
```

$$
\begin{array}{l}
genFill :: [Clause] \quad \text{-- cf. } F_{\text{Fill}} \\
genFill = \\
\quad [(-(encode\ (x, y, 0))) : [0] \\
\quad\ \mid x \leftarrow [0 \mathinner{..} sdksize], y \leftarrow [0 \mathinner{..} sdksize], (x > 0 \vee y > 0)]
\end{array}
$$

Makefile

```
SHELL := bash # ;(

all: solution
	sed -e 's/-[0-9]\{1,3\}//g' \
		-e 's/ \+/ /g' -e 's/0/TEH END\./g' \
		-e 's/ [0-9]1/\n/'g -e's/[0-9][0-9]//g' $<

solution: input
	-minisat $< $@

input: sudoku
	./$< > $@

sudoku: sdk.lhs
	ghc --make $< -o $@

.PHONY: clean
clean:
	rm -Rf solution input sudoku *.hi
```

Sample Output

```
% make clean all
rm -Rf solution input sudoku *.hi
ghc --make sdk.lhs -o sudoku
[1 of 1] Compiling Main             ( sdk.lhs, sdk.o )
Linking sudoku ...
./sudoku > input
minisat input solution
WARNING: for repeatability, setting FPU to use double precision
============================[ Problem Statistics ]=============================
|                                                                             |
|  Number of variables:          999                                          |
|  Number of clauses:           3159                                          |
|  Parse time:                  0.00 s                                        |
```

```
|                                                                             |
============================[ Search Statistics ]==============================
| Conflicts |          ORIGINAL         |          LEARNT          | Progress |
|           |    Vars  Clauses Literals |    Limit  Clauses Lit/Cl |          |
===============================================================================
|       100 |     675     2184     5544 |      800      100     16 | 32.433 % |
|       250 |     675     2184     5544 |      880      250     15 | 32.433 % |
|       475 |     675     2184     5544 |      968      475     17 | 32.433 % |
|       812 |     675     2184     5544 |     1065      812     18 | 32.433 % |
|      1318 |     675     2184     5544 |     1172     1318     18 | 32.433 % |
|      2077 |     675     2184     5544 |     1289     1313     20 | 32.433 % |
|      3216 |     675     2184     5544 |     1418     1634     19 | 32.433 % |
|      4924 |     675     2184     5544 |     1560     1566     17 | 32.433 % |
|      7486 |     672     2176     5528 |     1716     1230     15 | 32.733 % |
|     11330 |     639     1961     5098 |     1888     1813     17 | 36.036 % |
===============================================================================
restarts              : 63
conflicts             : 16339          (53868 /sec)
decisions             : 30812          (0.00 % random) (101585 /sec)
propagations          : 384474         (1267582 /sec)
conflict literals     : 292512         (10.26 % deleted)
Memory used           : 6.00 MB
CPU time              : 0.303313 s

SATISFIABLE
make: [solution] Error 10 (ignored)
sed -e 's/-[0-9][0-9][0-9]//g' -e 's/-[0-9][0-9]//g' \
-e 's/-[0-9]//g' -e 's/ \+/ /g' -e 's/0/TEH END\./g' \
-e 's/ [0-9]1/\n/'g -e's/[0-9][0-9]//g' solution
SAT

6 8 3 1 9 7 5 4 2
1 4 5 6 3 2 9 8 7
7 2 9 8 5 4 6 1 3
8 9 6 7 4 3 2 5 1
5 3 7 2 6 1 4 9 8
4 1 2 5 8 9 3 7 6
3 7 4 9 2 8 1 6 5
2 6 1 4 7 5 8 3 9
9 5 8 3 1 6 7 2 4 TEH END.
```