Introduction to SML
Example: A Tautology Checker

Michael R. Hansen

mrh@imm.dtu.dk

Informatics and Mathematical Modelling

Technical University of Denmark

Propositional Logic

A formula in propositional logic is either an atom given by its name which is a string, or a composite proposition built from atoms using the operators for negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$).

For example,

$$P \vee (Q \wedge \neg R)$$

is a formula of propositional logic, where P, Q and R are the atoms.

- define an SML datatype to represent formulas

Datatype

```
datatype prop = A of string
              | Neg of prop
              | And of prop * prop
              | Or of prop * prop

infix 3 Or
infix 4 And
```

Negation Normal Form

A formula is in *negation normal form* if the negation operator only appears as applied directly to atoms.

Write an SML function transforming a formula into an equivalent formula in negation normal form, using the de Morgan laws:

$$\neg(p \wedge q) \quad \Leftrightarrow \quad (\neg p) \vee (\neg q)$$
$$\neg(p \vee q) \quad \Leftrightarrow \quad (\neg p) \wedge (\neg q)$$

and the law: $\neg(\neg p) \Leftrightarrow p$.

Negation Normal Form: function

```
fun nnf(Neg(p1 And p2)) = nnf(Neg p1) Or nnf(Neg p2)
  | nnf(Neg(p1 Or p2))  = nnf(Neg p1) And nnf(Neg p2)
  | nnf(Neg (Neg p1))   = nnf p1
  | nnf(p1 And p2)      = nnf p1 And nnf p2
  | nnf(p1 Or p2)       = nnf p1 Or nnf p2
  | nnf p               = p
```

Conjunctive Normal Form

A *literal* is an atom or its negation.

A formula is in *conjunctive normal form* if it is a conjunction of formulas, where each conjunct (i.e. formula in the conjunction) is a disjunction of literals.

Write an SML function which transforms a formula into an equivalent formula in conjunctive normal form, using the above result and the laws:

$$p \vee (q \wedge r) \quad \Leftrightarrow \quad (p \vee q) \wedge (p \vee r)$$
$$(p \wedge q) \vee r \quad \Leftrightarrow \quad (p \vee r) \wedge (q \vee r)$$

From NNF to CNF

```
fun cnf(p And q)  = cnf p And cnf q

  | cnf(p1 Or q1) =
      (case (cnf p1, cnf q1) of

            (p', q' And r') => cnf(p' Or q')
                               And cnf(p' Or r')

          | (p' And q', r') => cnf(p' Or r')
                               And cnf(q' Or r')

          | (p',q')         => p' Or q'

  | cnf l           = l
```

Tautology

A formula ϕ is called a *tautology* if it is true for any assignment of truth values to its atoms.

- A disjunction of literals, a clause, is a tautology exactly when it contains both A and $\neg A$, for some atom A.
- A conjunction is a tautology precisely when each conjunct is a tautology.

Write a tautology checker in SML, i.e. an SML function which determines whether a formula is a tautology or not.

Tautology Checker

Assume function

```
isClauseTaut: prop -> bool
```

is given

```
fun isCnfTaut(p And q) = isCnfTaut p
                         andalso isCnfTaut q

  | isCnfTaut p        = isClauseTaut p


fun isTautologi f = isCnfTaut(cnf(nnf f));
```

Tautology of Clauses

Declare functions:

```
positiveAtoms: prop -> string list

negativeAtoms: prop -> string list

intersection: string list * string list -> string list

fun isClauseTaut c =
   let val posAtoms = positiveAtoms c
       val negAtoms = negaviteAtoms c
   in intersection(posAtoms, negAtoms) <> []
   end
```