CLP(B):
Constraint Logic Programming over Boolean Variables

```
:- use_module(library(clpb)).
```

1 Introduction

This library provides CLP(B), Constraint Logic Programming over Boolean variables. It can be used to model and solve combinatorial problems such as verification, allocation and covering tasks.

CLP(B) is an instance of the general CLP(·) scheme, extending logic programming with reasoning over specialised domains.

The implementation is based on reduced and ordered Binary Decision Diagrams (BDDs).

2 Boolean expressions

A *Boolean expression* is one of:

`0`	**false**
`1`	**true**
variable	unknown truth value
atom	universally quantified variable
~ *Expr*	logical NOT
Expr + *Expr*	logical OR
Expr * *Expr*	logical AND
Expr # *Expr*	exclusive OR
Var ^ *Expr*	existential quantification
Expr =:= *Expr*	equality
Expr =\= *Expr*	disequality (same as #)
Expr =< *Expr*	less or equal (implication)
Expr >= *Expr*	greater or equal
Expr < *Expr*	less than
Expr > *Expr*	greater than
`card(Is,Exprs)`	*see below*
`+(Exprs)`	*see below*
`*(Exprs)`	*see below*

where *Expr* again denotes a Boolean expression.

The Boolean expression `card(Is,Exprs)` is true iff the number of true expressions in the list *Exprs* is a member of the list *Is* of integers and integer ranges of the form `From-To`.

`+(Exprs)` and `*(Exprs)` denote, respectively, the disjunction and conjunction of all elements in the list *Exprs* of Boolean expressions.

Atoms denote parametric values that are universally quantified. All universal quantifiers appear implicitly in front of the entire expression. In residual goals, universally quantified variables always appear on the right-hand side of equations. Therefore, they can be used to express functional dependencies on input variables.

3 Interface predicates

sat(+*Expr***)** *[semidet]*

True iff *Expr* is a satisfiable Boolean expression.

taut(+*Expr, -T***)** *[semidet]*

Succeeds with $T = 0$ if the Boolean expression *Expr* cannot be satisfied, and with $T = 1$ if *Expr* is always true with respect to the current constraints. Fails otherwise.

labeling(+*Vs***)** *[multi]*

Assigns truth values to the Boolean variables *Vs* such that all stated constraints are satisfied.

sat_count(+*Expr, -N***)** *[det]*

N is the number of different assignments of truth values to the variables in the Boolean expression *Expr*, such that *Expr* is true and all posted constraints are satisfiable.

weighted_maximum(+*Weights, +Vs, -Maximum***)** *[multi]*

Maximize a linear objective function over Boolean variables *Vs* with integer coefficients *Weights*. This predicate assigns 0 and 1 to the variables in *Vs* such that all stated constraints are satisfied, and *Maximum* is the maximum of $\sum w_i \cdot v_i$ over all admissible assignments. On backtracking, all admissible assignments that attain the optimum are generated.

This predicate can also be used to *minimize* a linear Boolean program, since negative integers can appear in *Weights*.