CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

DECISION ANALYTICS.

Lab07: Constraint Propagation – AC3

BACKGROUND.

The AC3 algorithms allows to tighten the domains of a constraint network so that it is arc consistent. In this exercise you will implement a representation for Boolean Satisfiability problems and the AC3 algorithm to enforce arc consistency.

Task 1.

Write a Python class that models a Boolean variable for a constraint satisfaction problem. The class should store a name as string and the current domain with two Booleans indicating if True is part of the domain and if False is part of the domain.

The domain $D = \{T, F\}$ should be represented by [True,True], the domain $D = \{T\}$ should be represented by [False,True], the domain $D = \{F\}$ should be represented by [True,False] and the domain $D = \{\}$ should be represented by [False,False].

The class should have a method `GetValues()` for getting all possible values from the domain as well as a function `RemoveFromDomain(v)` for removing a value from the domain.

Task 2.

Write a Python class that models a Boolean disjunction constraint. It should be created by passing a list of positive and negative variables to the constructor. The class should store the scheme, i.e. the variables it operates on, and all combinations of assignments that are valid for the constraint as list of lists of positive variables (hint: use the function `itertools.combinations`).

For example the constraint with positive variables [x1, x2] and negative variable [v3] should represent the constraint $x_1 \vee x_2 \vee \neg x_3$ and will result in a scheme [x1,x2,x3]. The list of valid assignments is then [[],[x1], [x2], [x1,x2],[x1,x3], [x2,x3], [x1,x1,x3]]

The class should also contain a method `CheckConstraint(x,v)` for checking if a variable assignment $x_i = v$ could satisfy the constraint taking into account the domains of all other variables in the scheme of the constraint.

For example, checking (x1,True) should result in True, unless True has been removed from the domain of x1 by calling the function `x1.RemoveFromDomain(True)`.

Task 3.

Implement a model class to which you can add variables as defined in task 1 and constraints as defined in task 2. Implement the AC3 algorithm to make all domains of all variables in the model arc consistent.

(For the purpose of using this code in a future lab exercise, please make sure that you determine equality of variables based on their names instead of their object references, i.e. for two variables x1 and x2, do not compare them with x1==x2, but rather compare them with x1.name==x2.name)