CLab# and CaSPer: User Manual

Torbjørn Meistad, Yngve Raudberget and Geir-Tore Lindsve

September 2006

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S

Contents

Chapter 1

Introduction

This document describes version 1.0 of CLab# and version 1.0 of CaSPer. CLab# is a C# software tool for fast backtrack-free and complete interactive product configuration using constraint programming or binary decision diagrams. CaSPer is a C# library for solving CSP problems with CSP algorithms.

- Chapter 2 describes the CLab# library,
- Chapter 3 describes the CaSPer library which is used for handling CSP,
- Chapter 4 describes CLab# Configurator which is the client GUI application we have developed to demonstrate usage of the software,
- Chapter 5 describes the third party requirements for CLab#,
- Chapter 6 to 8 is API references for CLab#, CaSPer and the GUI.

1.0.1 Installing instructions

For all official versions of this thesis we enclose the project source code, binary files and web based API documentation on a supplementary CD. Everything can also be downloaded as a zip file from:

`http://tihlde.org/~torbjorm/clab/`

All source code for CLab#, CaSPer and CLab# Configurator is available in the *source* directory. Visual studio 2005 and .NET 2.0 has been used as the development environment. To open all parts use the solution file of the project: *ClabSharp.sln*. If the solution is built using a Visual Studio version supporting .NET 2.0, binary files should be created. Two test projects are included, *ClabSharpTest* and *CasperTest*. Both are console applications which gives a preview of the usage and possibilities of the CLab# and CaSPer libraries.

The *binary* directory contains compiled binary files of the source code, ready to be used. *.NET Framework 2.0* needs to be installed to be able to execute them. The *CLab# Configurator* can be run by executing the *ClabGui.exe* file.

The web based API documentation can be found in the *API* directory.

Chapter 2

CLab#

2.1 Overview

CLab# is an open source C# library for fast backtrack-free interactive product configuration. Two different solving techniques are implemented, and the user of the library can choose which one to use for configuration. The library is designed to seamlessly combine the two fundamentally different approaches of CSP and BDD in a single configurator tool. That means that irrespective of what approach is chosen, the usage of the library is the same. Both approaches share the same input data into the library and in the opposite side, they also share the same result data representation. What is really going on between the input and the output is hidden under the hood.

To make it easy to alter the code and implement new functions, each module has its own level of abstraction. The first level is the classes for general representation of the configuration problem and its resulting data. These objects are designed quite similar to the CP language definition, and does not offer any special functionality for solving the problem. All classes belonging to this level is in the `Clab.Data` namespace. At the next level, we have the two different solving approaches. For the *BDD* approach, we have the namespace `Clab.BDD` and for *CSP* the namespace `Clab.CSP`. Both namespaces has classes with the special data representations they need, derived from the general data representation. Both namespaces have a main interface class, which ties the functionality of each approach together, and offer methods for usage by the main interface class `Clab`. `Clab` again hides how two approaches work for the user by offering the methods needed for making an end user application for running product configuration.

When we perform a configuration problem with CLab#, an initial calculation of valid domains is calculated. This returns a list of variables and their valid domain values, in the same representation as in the problem definition. Now the user can select one of the valid domain values, and in that way make a partial configuration, knowing that the selected domain value leads to a full valid configuration. Then a new calculation of valid domains is run, which might lead to further reductions of the valid domains. For instance might a domain for a certain variable be reduced to contain only one value, and the partial configuration is extended. The user is for each iteration presented with fewer values to select from. Eventually there is only one solution left, and the problem is solved.

2.1.1 Main interface of CLab#

The `Clab` class is the main class of CLab. When `Clab` is initialized, it starts parsing the configuration problem. The `Clab` class has two constructors. One takes a XML file name as an argument, while the other one takes a memory stream with the XML file. Using either the XML file name or the memory stream, the internal configuration problem representation is made, and symbolically checked.

Using the `Clab` instance, it is now possible to initialize a problem solver. CLab# supports solving configuration problems using either *BDDs* or *CSP algorithms*. The initialization returns all domain values using the valid domains representation, which encodes the variables and their domain values as strings in the same form as in the XML file. Afterwards valid domain values can be computed reflecting the rules of the problem, using the chosen method. This operation returns the valid domain representation as well, which now might be reduced because of the rules. It is now possible for the user of the library to select a one of the valid domain values for a variable to further reduce the problem. Each time this operation is run, valid domain computation is performed and the results are returned. Because of this the configuration process is guaranteed to be backtrack-free. It is also possible to add an extra rule to the problem, which also returns the reduced valid domains. Adding extra rules can lead to a reduced (or solved) problem, or to a problem with no possible solutions.

When an error occurs, `Clab` throws a `ClabException` with an error message. This exception has to be handled by the user application.

Main public methods:

```
Clab.Clab(string fileName)
```

This is one of the constructors of `Clab`. It carries out the following operation:

- Parsing and type checking the CP problem XML file given in `XMLFileName`.

```
Clab.Clab(Stream XMLStream)
```

This is one of the constructors of `Clab`. It carries out the following operation:

- Parsing and type checking the CP problem XML file given as the stream `XMLStream`.

```
ValidDomains Clab.InitializeProblemSolver(CLabModus modus)
```

Initializes the problem solver for the specified modus. Returns all domain values in the configuration problem. The argument `modus` defines which problem solver to use. There are two options:

Method	Description
`bdd_modus`	Use *BDD*s to solve the problem
`csp_modus`	Use *CSP* algorithms to solve the problem

```
ValidDomains Clab.GetValidDomains()
```

Calculates the valid domains according to the rules of the problem and any rules added later in the process. Returns the valid domains.

```
ValidDomains Clab.GetValidDomainsExtraRule
(String variable, String selectedDomain)
```

Calculates the valid domains with the extended (or new) partial configuration. Returns the remaining valid domains.

```
ValidDomains Clab.GetValidDomainsExtraRule(Expression expr)
```

Calculates the valid domains according to the new rule. Returns the remaining valid domains.

```
int Clab.StatusUpdateCount()
```

Returns the maximum number of calls to the status event. Can For instance be used to set the max value of a progress bar.

```
int Clab.SetCSPVariableOrdering(CspVariableOrdering) varOrdering
```

Sets the variable ordering to be used when solving the problem with CSP algorithms. The argument `CspVariableOrdering` defines variable ordering to use. There are two options:

Method	Description
`vo_static`	Iterate through the variables according to their variable index.
`vo_minwidth`	Iterate through the variables according to their adjacency lists. Variable with the largest list is chosen first.

The default method is `vo_static`.

```
int Clab.SetBddCompileMethod(BddCompileMethod compileMethod)
```

The argument `compileMethod` defines the compilation approach. There are three options:

Method	Description
`cm_static`	Conjoin the BDDs of the rules in the order they appear in the CP file.
`cm_dynamic`	Add the BDDs of the rules to a work list. In each iteration, conjoin the two smallest BDDs and add the result to the work list. Return the resulting single BDD in the work list.
`cm_ascending`	Sort the BDDs of the rules according to their size. Conjoin the sorted BDDs from left to right.

The default method is `cm_dynamic`.

2.1.2 Valid Domains representation

Each variable and its valid domains are represented as a `ValidDomain` object. Both the variable and the domain values are represented as `Strings`. These objects are placed in a `ValidDomains` object, which contains a list of the `ValidDomain` objects. CLab# has an internal mapping from this string representation of variables and domain values, to the internal BDD or CSP representation. The application using the CLab# library does only need to know about a common representation, irrespective of what solution approach is chosen.

2.1.3 Expression structure

Expressions are constructed recursively in the same way as in the language definition, explained in section 2.2.1. Figure 2.1 shows the different classes which together are used to build any kind of expression supported by CLab# . All implementation classes implements the abstract Expressions class, and when they are initialized, a `ExprType enum` value has to be set. `ExpressionId` is used for representing a variable or a enumeration constant, and the enum value should be set to *et_id*. `ExpressionInt` should as the name implies be used for integer values, and the enum value should be set to *et_val*. To negate or invert another `Expression`, the `ExpressionNotNeg` class is used. Here the enum value tells whether to invert or to negate, by setting it to either *et_not* or *et_neg*. The `ExpressionBinary` class is used when we need to do a binary operation on two `Expressions`, and the enum value represents the operator.

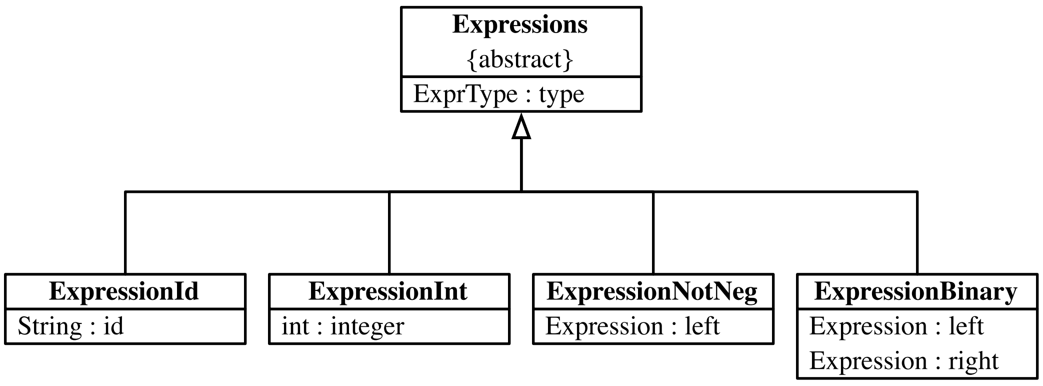


Figure 2.1: The figure shows the recursive expression structure of CLab#.

2.1.4 Status messages

CLab# supports status events, which for instance can be used to run a progress bar. All event handler functions which have been added to the status event (`StatusEvent`), is called with an int value representing the currently run operation, when an event occurs. The `StatusUpdateCount()` method can be used to find out how many updates can be expected for a search. A delegate is used as an event handler, and defines the interface for the event handler function which can be added. The delegate is called `StatusEventHandler`. When CSP algorithms are used to search for valid domains, the status event is updated each time the search is started for a new variable. For the BDD approach, the status event is updated each time a new expression is built, and for each of the conjunction operations on the resulting BDDs.

The status event delegate defines the event handler function interface `void MethodName(int FieldName)`. To add the method, the operation `StatusEvent += new StatusEventHandler(MethodName)` is run. ($+=$ is used to add, and $-=$ to remove.)

2.1.5 CLab# Code Example

In this section we present a small code example on how to use CLab#. We have not included class and namespace definitions, nor the specification of the namespaces and DLLs we rely on. A complete working copy can be found in the source code in the CLabSharpTest-project.

```
/* A simple example of using the CLabSharp library, using the printer
* example file.
*/
//Loading the printerExample.xml file: Change the path to match yours
String xmlFile = "printerExample.xml";
Console.WriteLine("Starting Clab#...");
//Start a Clab instance from the XML file.
Clab clab = new Clab(xmlFile);

//*********Problem solving using BDD:*********
clab.InitializeProblemSolver(CLabModus.bdd_modus);

//***********Problem solving using CSP*********
//(just uncomment the line below:)
//clab.InitializeProblemSolver(CLabModus.csp_modus);

//BDD compile method, :
```

```
clab.SetBddCompileMethod(BddCompileMethod.cm_ascending);

//This line prints out the progress of the search
//clab.StatusEvent += new Clab.StatusEventHandler(ctCSP.PrintStatus);

//Adding the rule ' "User" == "Visitor" ' manually can be done like this:
CLab.Data.ExpressionBinary binaryExpr =
new CLab.Data.ExpressionBinary(
    new CLab.Data.ExpressionId("User"),
    Common.ExprType.et_eq,
    new CLab.Data.ExpressionId("Visitor"));


//Adding the created rule to the CP-structure:
clab.Cp.AddRule(binaryExpr);

//Start the Valid Domains Computation,
//independent of the choice of CSP or BDD approach
ValidDomains vd = clab.GetValidDomains();


Console.WriteLine("\n\n");
Console.WriteLine("\n\nValid domains after adding user" );
Console.Write("\"Papersize\" == \"A4\":\n");

//Simulating the interactive selection during a
//configuration session, we select "Papersize" == "A4".
vd = clab.GetValidDomainsExtraRule("Papersize", "A4");

//Print out of the remaining valid domains.
for (int i = 0; i < vd.ValidDoms.Count; i++)
{
    ValidDomain v = (ValidDomain)vd.ValidDoms[i];
    Console.Write("\n" + v.VarName + ":");
    for (int j = 0; j < v.Domains.Count; j++)
    {
        Console.Write(" " + v.Domains[j]);
    }
}
```

2.2 Configuration Language Definition

CLab# supports two different input languages, which both compile to the same internal data structure. Due to the nature of being derived from CLab 1.0, CLab# supports the same CP language as CLab 1.0 with some alterations related to string identifiers. This language is defined in Section 2.2.1 In addition, we have developed a XML structure which can be used to store a configuration problem. Since XML is a cross platform language we considered it to be a good way of making the CP-language accessible to a wide variety of systems and platforms. The XML structure is defined in Section 2.2.2

2.2.1 CP Language Definition

The CP language has two basic types: *range* and *enumeration*. A range is a consecutive and finite sequence of integers. An enumeration is a finite set of strings. The Boolean type is the range from 0 to 1. Range and enumeration types can be defined by the user. A CP description consists of a *type declaration*, a *variable declaration*, and a *rule declaration*. The type declaration is optional if no range or enumeration types are defined:

```
cp       ::= [ type {typedecl} ] variable {vardecl} rule {ruledecl}

typedecl ::= id [ integer . . integer ] ;
         | id { idlst } ;

vardecl  ::= vartype idlst ;

vartype  ::= bool
         | id

idlst    ::= id {, idlst}

ruledecl ::= exp ;
```

The identifier in CLab# has been altered from the defined identifier in CLab 1.0 to support strings identifiers that start with numbers. An identifier is hence a sequence of numbers, letters, underscore and the character ” ”, or a string enclosed with the character `"`. An integer is a sequence of digits possibly preceded by a minus sign. The symbol `//` start a comment that extends until the end of the line. The only comments which are stored in the XML format (see Section 2.2.2), is comments which explicitly describes the file with:

```
// Description: <text>
// Author: <text>
// Date: YYYY-MM-DD
```

The syntax of expressions is as follows:

```
exp ::= integer
    | id
    | - exp
    | ! exp
    | ( exp )
    | exp op exp

op ::= * | / | % | + | -
   | == | !=
   | < | > | <= | >=
   | && | & | || | | | >>
```

The semantics, associativity, and precedence of arithmetic, logical, and relational operators are defined as in C/C++. Hence, `!`, `/`, `%`, `==`, `!=`, `&&`, and `||` denote logical negation, division, modulus, equality, inequality, conjunction, and disjunction, respectively. The only exception is the pipe operator `>>` that denotes implication. The precedence and associativity is shown in Table 1. Notice that the convention of following C/C++ precedence causes the pipe operator to have higher precedence than is usual for logical implication.

Operators	Associativity
`! -`	right to left
`* / %`	left to right
`+ -`	left to right
`>>`	left to right
`< <= > >=`	left to right
`== !=`	left to right
`&&`	left to right
`||`	left to right

Table 2.1: Precedence and associativity of operators

The semantics of an expression is the set of variable assignments that satisfy the expression. For example assume that the type of variable x and y is the range `[4..8]`. The set of assignments to x and y that satisfies the expression `x + 2 == y` is then $\{\langle 4,6\rangle, \langle 5,7\rangle\}$. An assignment for which there exists an undefined operator in the expression is assumed not to satisfy the expression. Thus, the set of assignments to x and y that satisfies `x / y == 2` is $\{8,4\}$. Conversion between Booleans and integers is also defined as in C/C++. True and false is converted to 1 and 0, and any non-zero arithmetic expression is converted to true. Due to these conversion rules, it is natural to represent the Boolean constants true and false with the integers 1 and 0.

The CP file for the printer example can be seen below:

```
// Description:  CLab# 1.0 Example
// Author:  CLab# Crew
// Date:  2006-08-16

type
  userType {Visitor,Employee};
  paperType {A3,A4,A5};
  printerType {Simple,Advanced};
  inkType {Color,Black};

variable
  userType User;
  paperType Papersize;
  printerType Printer;
  inkType Ink;

rule
  ((Printer == Simple) >> (Papersize != A3));
  ((Printer == Simple) >> (Ink == Black));
  ((Papersize == A3) >> (Ink == Black));
  ((User == Visitor) >> (Printer == Simple));
```

2.2.2 XML Language Definition

The Extensible Markup Language (XML) is a W3C-recommended general-purpose markup language for creating special-purpose markup languages, and can be used to both describe and contain data. XML is ideal for documents containing structured information, where the information can contain both content and a definition of what role that content plays.

The XML data is structured as a tree with elements, and the entire tree structure is called a document. XML has no data description separate from the data itself, unlike fixed or delimited data formats. The documents are self-describing. The data has specially formatted tags around it that give the data a name as well as a position in the document's tree structure. We can see from this that XML as a document format is ideal for storing object structures in a structure and well-defined way, which is the main decision for choosing it for CLab#.

The XML schema we have defined for CLab# can be shown in simplified form in Figure 2.2. The solid lines show connections to inner elements, and dotted lines shows the recursive structure in rules.

The XML structure has in addition to a header element, three main elements (`type`,

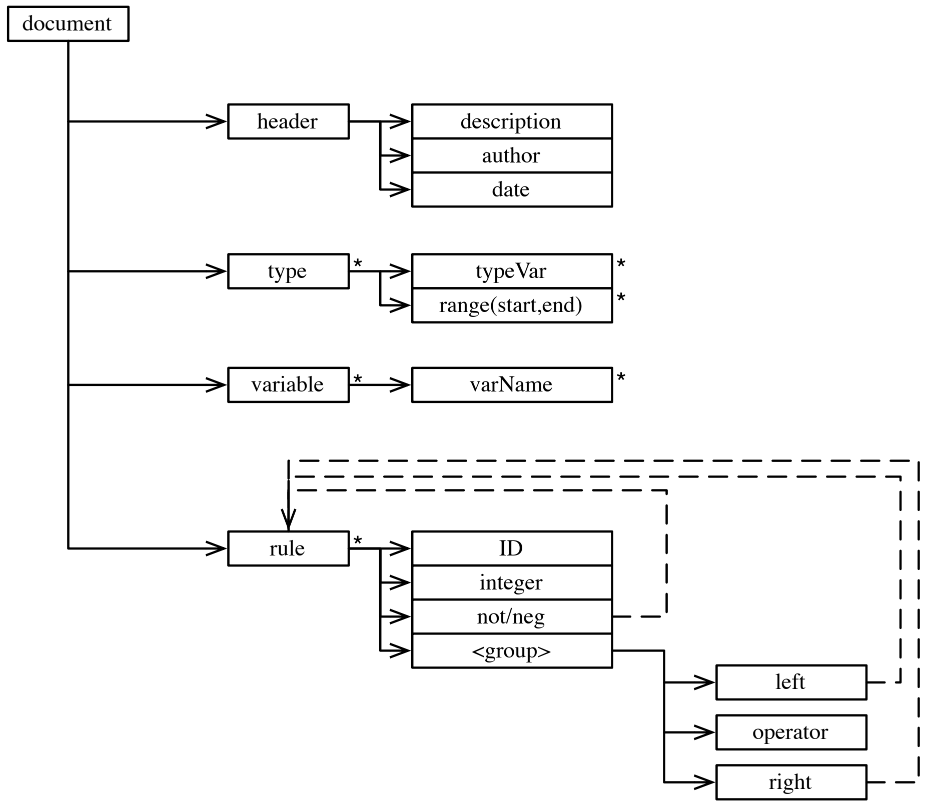


Figure 2.2: Simplified XML Schema for the CLab# document format. Note that $\star$ denotes entities which can occur several times. Dotted lines denote recursive connections

`variable`, `rule`) which each explicitly denotes the same declarations as in the CP language. The `header` element is used for storing meta data for describing the problem configuration:

```
<header>
    <description>Printer Example</description>
    <author>Torbjorn Meistad, Yngve Raudberget and Geir-Tore Lindsve</author>
    <date>2006-08-16</date>
</header>
```

The `type` element is used for declaring the variable type to be used in the configuration. It consists of a single `typeDecl` element (not shown in Figure 2.2) for each type declaration. The `typeDecl` element can consist of either one or more `typeVar` elements or a `range` element. Former is the enumerator type equal to the identifier in the CP language and can consist of letters, numbers, underscore and the character " ". A range is declared with the attributes `start` and `end`, each can consist of a sequence of digits possibly preceded by a minus sign:

```
<typeDecl
    name="UserType">
    <typeVar>Visitor</typeVar>
    <typeVar>Employee</typeVar>
</typeDecl>
<typeDecl
    name="ARange">
    <range start="0" end="10">
</typeDecl>
```

The `variabel` element is used for declaring the variables to be used in the configuration. It consists of a single `varDecl` element (not shown in Figure 2.2) for each variable declaration. The `varDecl` element can consist of one or more `varName` elements, each representing a variable of that type. Which type to use is an attribute (`varType`) to the `varDecl` element:

```
<varDecl
varType="PrinterType">
    <varName>Printer</varName>
    <varName>AnotherPrinter</varName>
</varDecl>
```

The `rule` element is used for declaring the rules, or constraints, to be used in the configuration. The `rule` element can consist of one or more `ruleDecl` elements, each denoting a single rule. The elements `left`, `right`, `not`, `neg` are all of the same type and can be used recursively. Negation (!) can be set by enclosing the expression to negate in a `not` element, and inversion (-) can be set by enclosing the expression to invert in a `neg` element. The operators which can be used in the `op` element is `&, &&, |, ||, <, <=, >, >=, !=, ==, >>, +, -, *, /, %`

An example of the `ruleDecl` element:

```
<ruleDecl>
    <left>
        <left>
            <id>Printer</id>
        </left>
        <operator>==</operator>
        <right>
            <id>Simple</id>
        </right>
    </left>
    <operator>>></operator>
    <right>
        <left>
            <id>Papersize</id>
        </left>
        <operator>!=</operator>
        <right>
            <id>A3</id>
        </right>
    </right>
</ruleDecl>
```

2.3 Parsing

There are two parsers implemented in CLab#. One which parses XML input and injects the data in the internal data structure, and another which parses plain CP text to XML. In addition there is a `CLabTextParser` class which is a helper class for converting XML to plain text and XML to plain text CP.

2.3.1 ClabXMLParser

To parse XML input data to objects in the internal data structure, we use standard classes from the `System.Xml` and `System.Xml.Schema` namespaces. A `XmlReader` is set up to traverse the complete XML structure and inject the input as objects in the internal data structure.

The public methods available from the `ClabXMLParser` class is:

```
void ClabXmlParser(CP cpObject)
```

This is the constructor for the XML parser. The input is the CP object which will hold the information from the XML file.

```
void InitializeParser()
```

This method is called internally. It carries out the following operation:

- Sets up the XML validation service which validates the input data against the XML schema defined in CLab#.

```
void Parse(Stream stream)
```

Parses the specified stream. It carries out the following operation:

- Sets up a new reader object which parses XML input from the specified stream.

```
void Parse(String XMLUrl)
```

Parses the specified XML file. It carries out the following operation:

- Sets up a new reader object which parses XML input from the specified file location.

The `Parse` methods call the internal method `RunParse()` which is the main parse method. When it hit start elements, attributes are stored and records to keep track of where the parser is in the document structure are updated. Most of the data injection is done when the parser hit a corresponding end element. It checks to see what type of element it is and the action taken depends on that and its content. Since the traversal of the document structure is done from the top to bottom, we cannot inject data before we hit an end element. To facilitate that, we store the data in stacks until it is time to store it to the CP object.

The ClabXMLParser throws and `ClabXMLParserException` when it encounters an error.

2.3.2 TokenParser

The token based parser used for parsing plain CP text to XML is implemented with the use of the free GOLD Parsing System[1]. GOLD uses the LALR algorithm to analyze syntax and a *Deterministic Finite Automaton* to identify different classes of tokens. The grammar file which the parser uses are based on the same grammar file used in CLab 1.0[2], and it is compiled into tables which are used in runtime parsing. As the parser accepts a sequence of tokens, it determines, based on the compiled tables, when the grammars' respective rules are complete and verifies the syntactic correctness of the token sequence. The end result of the process is a *derivation* which represents the token sequence organized following the rules of the grammar.

The `TokenParser` class implements a parser which, depending on the accepted token, takes the required actions to write the input to a specified XML stream. It uses a standard XmlWriter from the `System.Xml` namespace.

2.3.3 CLabTextParser

As previously stated, `ClabTextParser` is a helper class used by our GUI application. It is developed so that it is trivial to move data between the textual user interface and the XML files.

These are the public methods in the `ClabTextParser`:

```
String XMLtoCP(string filename)
```

Parses a XML file and prints the data to a string and returns it. It carries out the following operations:

- Creates a new CP object from the specified filename. The `CP` class then uses the `ClabXMLParser` to get the data from the XML file.
- Prints the contents of the CP structure to a string with the use of the `toString()` method for the content objects, and returns that string.

[1]GOLD Parsing System: http://www.devincook.com/goldparser/

[2]The grammar is the same as in CLab 1.0, with the addition to the identifier so that strings enclosed with the character " and identifiers which starts with a number is accepted.

```
MemoryStream CPtoXML(string input)
```

Parses the input string and returns a stream with the XML representation of the input data. It carries out the following operations:

- Creates a new stream to write the XML data to.
- Checking to see if the input contains header data. This must be handled outside of the `TokenParser`, since the latter ignores commented lines. If header data is found, proper header elements are created in the XML stream.
- Sends the input data to the `TokenParser` with the stream as an argument.
- After the `TokenParser` is done, the stream is returned.

2.4 Error handling

To facilitate an easy error handling, CLab# throws a single expression type, `ClabException`. There are some other expression types defined and used internally:

- `ClabInternalErrorException`
- `ClabInvalidExpressionException`
- `ClabSymbolException`
- `ClabXMLParserException`
- `TokenParserException`

Currently, the different exceptions are handled the same way, they are all enclosed in an exception object of type `ClabException`, including all system exceptions. The reason for separating the different exceptions is to be able to know exactly what part of CLab# failed, and to be able to act differently according to that. Since they are all enclosed in a `ClabException`, a client application is only required to catch this exception type for error handling. If the client application requires different measures taken for different types of error, it is trivial to collect the inner exceptions and act upon them. The `message` property of the `ClabException` object contains the error message from the underlying exception, so it is strictly not needed to go in the inner exceptions. All the custom exception classes in CLab# are located in the `Clab.Exceptions` namespace.

Chapter 3

CaSPer

3.1 Overview

CaSPer is a library for solving CSP problems, using CSP algorithms. Currently only one algorithm is implemented, but the library is designed to easily support other algorithms. The library consists primary of two parts. The first part is the general data representation part of a CSP problem, while the other is the algorithmic part.

The data is represented as variables with domain data as `CasperVarDom` objects, and expressions as `CSPExpr` objects. Another optional data type is the constraint graph, represented as a `ConstraintGraph` object with multiple `AdjacencyList` objects inside.

The `CasperVarDom` class is designed to be small and easy, with a variable ID and a list of ints representing the domain values. The variables should be encoded with subsequent int values as IDs, starting from 0. This design should make it possible to encode any kinds of variables and domain values. If the algorithms needs special structures, these can easily be build from this representation. The expression and constraint graph representations are described in detail later.

The library is designed not to be dependent of a specific algorithm. Therefore the algorithmic part is loosely connected to the data representation part of the library. This is done to allow other algorithms to be added, using the same basic data, and to allow the usage of the implemented algorithm without using all the other parts of the library. The current implementation contains *Generalized Look Ahead* with *Select Value Forward Checking*.

3.1.1 Main Interface class of CaSPer

The `CSP` class is the main class of the `CaSPer` library. Variables and domain data, expressions and optionally the constraint graph, is the basic data needed for setting up the library for solving a problem. The class has to main methods, one for calculation of the problems' valid domains, and one for reducing the problem with an user selected value. When an error occurs, the `CSP` class throws a `CasperException` with an error message. This exception has to be handled by the user application.

Main public methods:

```
Casper.CSP
(CSPExpressions expressions, List<CasperVarDom> casperVarDoms)
```

This is one of the constructors of `CSP`. It carries out the following operations:

- Initializing the class with the input data.
- Making a mapping from variable ID to internal index for all variables in `casperVarDoms`.

```
Casper.CSP(CSPExpressions expressions)
```

This is one of the constructors of `CSP`. If this constructor is used variables and domain data has to be added afterwards, before using any search functionality. It carries out the following operation:

- Initializing the class with the input data.

```
Casper.CSP()
```

This is one of the constructors of `CSP`. If this empty constructor is used both variables and domain data, and expressions have do be added afterwards, before using the search functionality.

```
ConstraintGraph Casper.ConstraintGraph Set/Get property
```

This property allows setting or getting the optional constraint graph. The constraint graph is needed for certain variable orderings, like minimum width ordering.

```
void Casper.SetVariableOrdering(CspVariableOrdering varorder)
```

Sets the variable ordering used by the searching algorithm The argument `varorder` defines which variable ordering to use. There are two options:

Method	Description
`vo_static`	Iterate through the variables according to their variable index.
`vo_minwidth`	Iterate through the variables according to their adjacency lists. Variable with the largest list is chosen first.

Default variable ordering is *static*.

```
CSPExpressions Casper.Expressions Set/Get property
```

This property allows setting or getting the expressions data.

```
Boolean Casper.OmitTestOnSingleValuedDomains Set/Get property
```

This property allows setting or getting the switch for omitting searches on variables with single valued domain. That is variables which are part of a partial configuration.

Default is *false*.

```
int Casper.AddCasperVarDom(CasperVarDom vardom)
```

This method adds `vardom` to the list over variables and domain data, and updates the internal mapping. The int returned is the internal index it gets inside *CaSPer*.

```
List<CasperVarDom> Casper.ValidDomainsUserChoice
(CasperVarDom var, int domainVal)
```

This method runs the valid domains method, after reducing the `var` object to only have `domainVal` as valid domain value. That is adding `domainVal` to the partial configuration. The valid domains results are returned as a list of variables and domain data. `null` is returned if the problem for some reason is not valid. A reason could be passing in an invalid domain value or that `ValidDomains()` method is not run first.

```
List<CasperVarDom> Casper.ValidDomains()
```

This method runs the valid domains method. The valid domains results are returned as a list of variables and domain data, if the problem is valid. Otherwise `null` is returned.

3.1.2 Expression structure

Expressions are constructed recursively almost in the same way as in CLab#. Figure 3.1 shows the different classes which together are used to build the expressions CaSPer supports. There are three different abstract classes, where the top one is the common class representing all others. There are own implementation classes for the following expressions:

- Integer values, which should use the `CSPExprValueInt` class.
- Enumeration constants, which should use the `CSPExprValueConst` class.
- Boolean variables, which should use the `CSPExprVarBool` class.

- Enumeration variables, which should use the `CSPExprVarEnum` class.
- Integer variables, which should use the `CSPExprVarInt` class.
- Negated `CSPExpr` objects, which should use the `CSPExprNeg` class.
- Inverted `CSPExpr` objects, which should use the `CSPExprNot` class.
- `CSPExpr` objects with an operator, which should use the `CSPExprBin` class.

Each implementation class uses the double dispatch pattern towards the consistent implementation. This way each expression type has its own special method, and we avoids type testing.

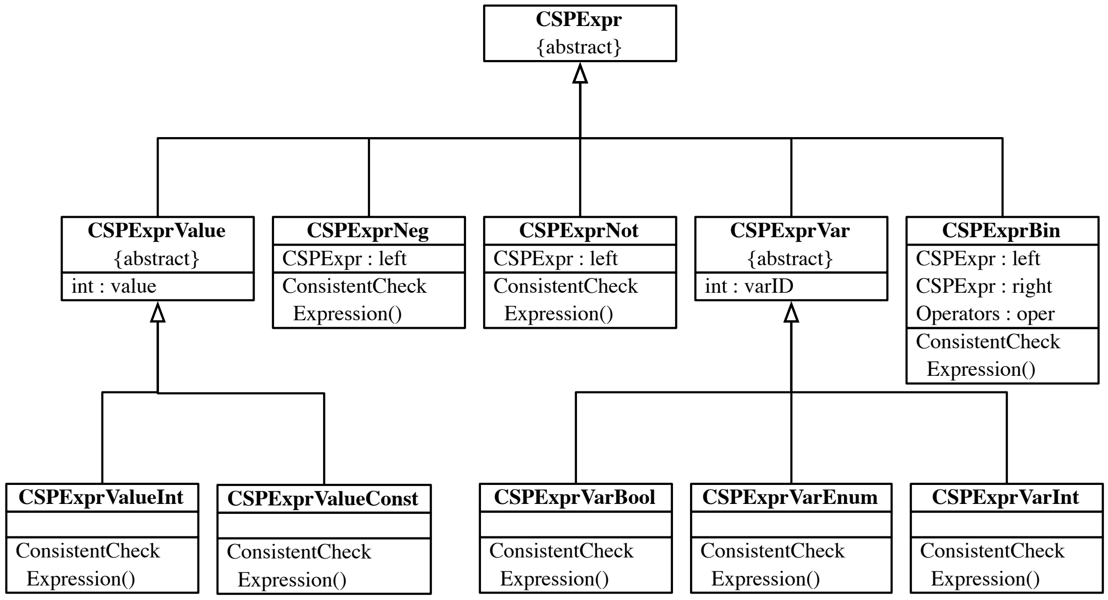


Figure 3.1: The figure shows the recursive expression structure of CaSPer.

Figure 3.2 shows how the different expressions for each rule is represented. Each expression is placed inside a `CSPExprWrapper` class, which also consists of a set of all the variable IDs within the expression. This set is used to determine whether an expression should be checked by the consistency method for a certain assignment of variables. The `Expressions` class keeps track of all these classes.

3.1.3 Status messages

CaSPer supports status events, and this function is used to tell CLab# when it should update its status. All event handler functions which have been added to the status event

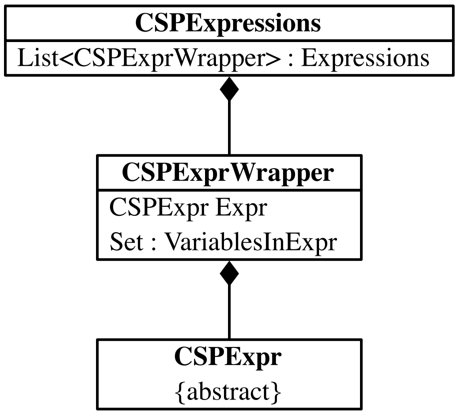


Figure 3.2: The figure shows how the expressions for all the rules are represented.

(`StatusEvent`), is called with an int value representing the currently run operation, when an event occurs. The `StatusUpdateCount()` method can be used to find out how many updates can be expected for a search. A delegate is used as an event handler, and defines the interface for the event handler function which can be added. The delegate is called `StatusEventHandler`. The status event is updated each time the search is started for a new variable in the `ValidDomains()` or the `ValidDomainsUserChoice()` methods.

The status event delegate defines the event handler function interface `void MethodName(int FieldName)`. To add the method, the operation
`StatusEvent += new StatusEventHandler(MethodName)` is run. ($+=$ is used to add, and $-=$ to remove.)

3.2 Error handling

As with CLab#, CaSPer throws a single expression type, `CasperException`, to facilitate an easy error handling. In addition to the times when CaSPer itself throws this exception, all system exceptions are enclosed in this exception type.

Since all exceptions thrown out of CaSPer are enclosed within a `CasperException` object, CLab# only needs to focus on catching this single exception type from CaSPer, and take the necessary measures. What CLab# does, is to again enclose this exception in a `ClabException` which is thrown out to the client application.

3.2.1 Casper Code Example

In this section we present a small code example on how to use the CaSPer library. We have not included class and namespace definitions, nor the specification of the namespaces and .dlls we rely on. A complete working copy can be found in the source code in the CasperTest-project.

```
/* A simple example of using the CaSPer Library:
* Consider the following constraint problem:
* variables x0, x1, x2
* Domain for all variables :{0,1}
* Rules:
* 1) x0 = x1;
* 2) x1 != x2;
*/

Csp casper = new Csp();

//Create domains 0 - 2 as a list of integers
List<int> domain0 = new List<int>();
List<int> domain1 = new List<int>();
List<int> domain2 = new List<int>();

//Add values 0,1 to all domains
domain0.Add(0);
domain0.Add(1);
domain1.Add(0);
domain1.Add(1);
domain2.Add(0);
domain2.Add(1);

//Create variable objects with variable ids 0,1,2 from
//the domains.
casper.AddCasperVarDom(new CasperVarDom(0, domain0));
casper.AddCasperVarDom(new CasperVarDom(1, domain1));
casper.AddCasperVarDom(new CasperVarDom(2, domain2));

//Make the rules as expression objects
casper.Expressions = new CSPExpressions();
CSPExpr expr1 = new CSPExprBin(new CSPExprVarInt(0),
 new CSPExprVarInt(1),StaticData.Operators.eq);
CSPExpr expr2 = new CSPExprBin(new CSPExprVarInt(0),
 new CSPExprVarInt(2), StaticData.Operators.gt);


//Make a set of which variables are found in which rule
```

```
Set variablesInRule1 = new HybridSet();
Set variablesInRule2 = new HybridSet();
variablesInRule1.Add(0);
variablesInRule1.Add(1);
variablesInRule2.Add(1);
variablesInRule2.Add(2);

//Creating a Constraint Graph is optional, but required if
//one wishes to use the minimum width ordering of variables.
ConstraintGraph constraintGraph = new ConstraintGraph();
//Add the variableInRule-sets into the constrintGraph as adjacency lists.
constraintGraph.UpdateAdjacencyList(variablesInRule1);
constraintGraph.UpdateAdjacencyList(variablesInRule2);

//*****variable ordering:******
//1. Add the constraint graph to the casper object:
casper.CGraph = constraintGraph;
//2. Set the variable ordering, static(default) or minimum width:
casper.SetVariableOrdering(Casper.CspVariableOrdering.vo_minwidth);


//Make an expression-wrapper that also include the
//set of variables that are found in the rule
casper.Expressions.AddWrappedExpr(new CSPExprWrapper(expr1, variablesInRule1));
casper.Expressions.AddWrappedExpr(new CSPExprWrapper(expr2, variablesInRule2));

//Calculate the valid domains of the problem.
//This runs the CSP search algorithm 'under the hood'.
List<CasperVarDom> validDoms = casper.ValidDomains();

//Print out of valid domain values:
for (int i = 0; i < validDoms.Count; i++){
    Console.Write("VarID: " + validDoms[i].VarID + ": ");
    for (int j = 0; j < validDoms[i].DomainValues.Count; j++)
         Console.Write(" " + validDoms[i].DomainValues[j] + " ");
    Console.WriteLine(""); }
```

Chapter 4

CLab# Configurator

This section presents an overall look at the functionality of the graphical user interface we have provided, and the use of threads in the application.

4.1 Overview

The graphical user interface provided, CLab# Configurator, have been developed to be an example implementation of a client application for the CLab# library. The application is developed with standard Windows Forms to look familiar to most Windows users. The application provides both an editor (*Problem Editor*) and an interface (*Set Parameters*) for executing a problem configuration and setting values for each variable. These are separated in two tabs in the interface. This section details the menu options available, the functionality of the editor and the functionality of the executing interface.

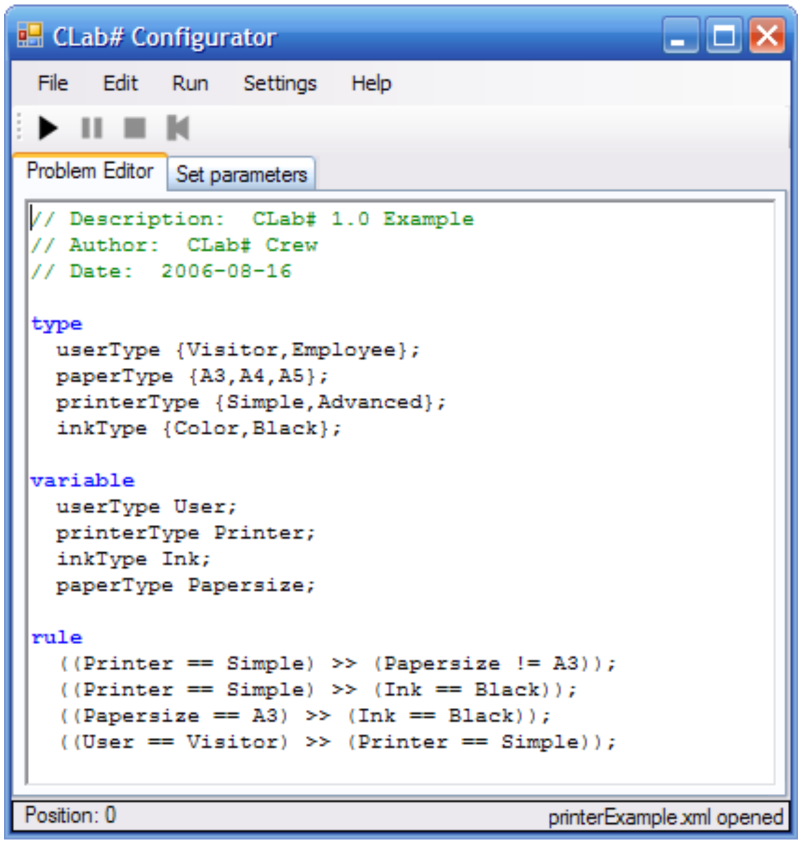
Menu options CLab# Configurator covers the necessary options one might expect from a editing environment. The *File menu* covers operations per file basis such as saving, opening and closing. It also contains the Exit options to quit the application. The *Edit menu* covers clipboard options (Cut, Copy, Paste). The *Run menu* contains menu equivalents to the buttons in the interface. These options are Start Search, Pause Search, Stop Search and Restart. The *Settings menu* is the interface to adjust the current parameters for the execution of a configuration problem. A user may choose to solve a configuration problem with either BDD or CSP. There are also several options available which only applies when using BDDs. These include the compile method to use (Static, Dynamic, Ascending), Setting initial DB Cache and number of BD nodes, and the Max Increase value. For CSP, we can adjust the variable ordering to either Static or Minimum Width. The final setting available is an option to turn autocompletion on or off. When enabled, the autocompletion feature will automatically enter closing characters as displayed in Figure 4.1.

```
(    ->    ()
{    ->    {};
[    ->    [];
\    ->    //
```

Figure 4.1: The autocompletion of delimiter characters

The final menu option is the *Help menu*, which

Problem Editor The Problem Editor is an editor for the configuration files used by CLab#, and can open/save both plain text CP files and XML files. A screenshot of the editor can be seen in Figure 4.2.

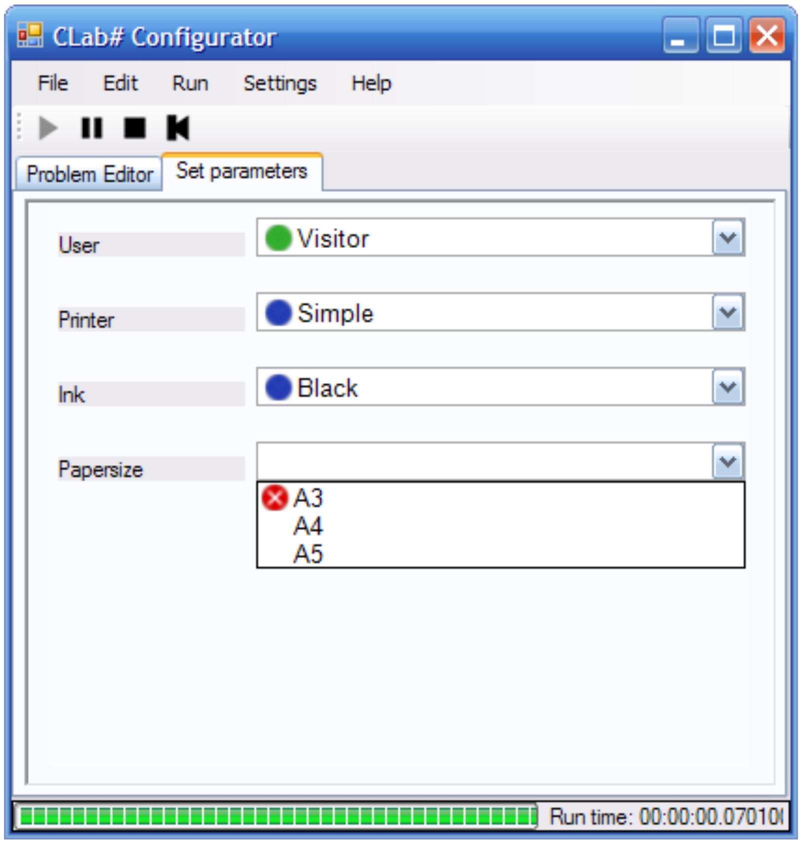


```
// Description:  CLab# 1.0 Example
// Author:  CLab# Crew
// Date:  2006-08-16

type
  userType {Visitor,Employee};
  paperType {A3,A4,A5};
  printerType {Simple,Advanced};
  inkType {Color,Black};

variable
  userType User;
  printerType Printer;
  inkType Ink;
  paperType Papersize;

rule
  ((Printer == Simple) >> (Papersize != A3));
  ((Printer == Simple) >> (Ink == Black));
  ((Papersize == A3) >> (Ink == Black));
  ((User == Visitor) >> (Printer == Simple));
```

Figure 4.2: The Problem Editor

The editor have built-in syntax highlighting for comments and the strings `type`, `variable`, `rule` and `bool`, which all are used for own purposes in the configuration file. In addition, it provides an option for autocompletion of the delimiter characters shown in Figure 4.1. Be aware though, that the implementation of the syntax highlighting has broken support for undo/redo, so the editor does not have any integrated support for these actions. This is due to parsing of the editing line when inserting text, and this is registered by the undo service in C#.

Set parameters The Set Parameters tab let users assign values to variables in the configuration file during execution. A screenshot of this tab can be seen in Figure 4.3.



User ● Visitor

Printer ● Simple

Ink ● Black

Papersize

A3
A4
A5

Run time: 00:00:00.07010

Figure 4.3: The Set Parameters tab

As we can see from the screenshot, this interface provides drop-down lists for each variable in the configuration. The list contains all values in the domain, although some have symbols denoting the availability of them. The symbols follows the structure presented in Table 4.1

At current state, CLab# does not support backtracking during configuration. This means that as soon as a value is chosen for a variable, then it cannot be changed to another value without restarting the configuration.

Symbol	Description
Green	The value is chosen by the user
Blue	The value is chosen by CLab# since it is the only available value in the domain
Grey	Values not chosen for a variable (when another value is chosen)
Red with white X	Unavailable due to the current constraints
None	Value is available

Table 4.1: Symbol structure for the Set Parameter tab in CLab# Configurator

Error messages CLab# Configurator provides a common interface for all error messages displayed. All exceptions which occur within CLab# are caught and displayed in the GUI. There is one exception which gets a special treatment when it comes to being presented, and that is an exception which occurs while saving a configuration file. If the contents of the editor differs from the defined language, then an error will be displayed telling what is wrong and give the user an option to move the cursor to the location of the error.

4.2 Threads

Since a search might take some time to carry through, the search is run in its own thread. If this is not done, the GUI would seem not to response. The search thread updates the GUI when it is finished with a search. Since *Windows Forms* is not thread safe[1], it is not allowed to get or set any property on a `Windows.Forms` Control, from another thread than the thread which handles the message queue (The GUI thread). To deal with this problem, we use the `InvokeRequired` property. If this property returns `true`, we make a call to the `Invoke` method, which can enqueue a delegate with the data needed to update the GUI. The delegate is enqueued into the message queue of the GUI control we want to update, and run by the GUI thread. Therefore the `MainGUI` class has a method for each kind of update which is run from another thread. We have also defined delegates for different kinds of input, to support the methods and their input data. By running the search in its own thread we get a more responsive GUI. We have also taken the advantage to make it possible to pause the search during a search. This may be useful when you need the computer for something else, since a search takes up as much of the CPU resources as it can.

The third party BDD package we use for solving the configuration problems with BDDs, *Buddy_sharp*, crashes if it is run more than once without restarting the GUI. This is because *Buddy_sharp* is not thread safe. We have chosen not to deal with this issue for

[1]Making Windows Forms thread safe - http://www.codeproject.com/csharp/threadsafeforms.asp

multiple reasons. If we run the search and GUI in the same thread, the issue is solved, but that leaves us in many cases with a solution which seems not to respond. We think of this solution as worse, since even if it crashes when the problem is run from scratch, it is possible to run one complete configuration with BDDs. It should be possible to make *Buddy_sharp* thread safe, and solve the issue completely, though this is not a part of this thesis.

Chapter 5

Third party requirements

The development of CLab# and CaSPer have been facilitated with some third party contributions. We have made sure that none of the contributions have licenses which conflicts with CLab# or each other, and are free to use.

For solving configuration problems using BDDs, we are using the *BuDDy* package, implemented by J. Lind-Nielsen in C++ and the *Buddy_sharp* C# wrapper for BuDDy, implemented by Rune Møller Jensen and Ken Friis Larsen.

We make extensive use of *sets* in our implementations, and since C# does not have any internal set implementation, we are using a publicly available Set implementation, available in the package `Iesi.Collections`[1].

The token parser which parses plain text to XML format has been developed with the free *GOLD Parsing System*. The parser uses a compiled grammar table which have been compiled from a modified grammar file from CLab 1.0 with an dedicated application provided by the parsing system.

[1]Iesi.Colletions - http://www.codeproject.com/csharp/sets.asp

Chapter 6

CLab# 1.0 API Reference

6.1 Namespace CLab

Namespace Contents *Page*

Interfaces

Classes

6.1.1 Classes

CLASS **Clab**

The interface class for the library. The library is designed in such way, that everything should be done through this class.

DECLARATION

```
public class Clab
    : Object
```

PROPERTIES

- *Cp*

  `public` **CLab.Data.CP Cp** `{ get; }`

  Gets the "CLab.Data.CP" problem instance.

- *Initbddnodes*

  `public` **int Initbddnodes** `{ get;  set; }`

  Gets or sets the initial number of bdd nodes.

- *Initdbcache*

  `public` **int Initdbcache** `{ get;  set; }`

  Gets or sets the initial size of db cache.

- *Maxincrease*

  `public` **int Maxincrease** `{ get;  set; }`

  Gets or sets the BDD max increase number.

CONSTRUCTORS

- *Constructor*

  `public` **Clab** `( )`

  Initializes a new instance of the ”Clab” class.

  - **Parameters**
    - `fileName` - Name of the csp problem file.

- *Constructor*

  `public` **Clab** `( )`

  Initializes a new instance of the ”Clab” class.

  - **Parameters**
    - `xmlStream` - The cp file stream.

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *GetUnsatisfiableRule*

  `public` **string GetUnsatisfiableRule**`( )`

  Gets the unsatisfiable rule, if BDD modus is chosen. This information is not possible to retrieve in CSP modus.

- *GetValidDomains*

  `public` **CLab.ValidDomains GetValidDomains**`( )`

  Gets the valid domains for the constraint problem.

- *GetValidDomainsExtraRule*

  `public` **CLab.ValidDomains GetValidDomainsExtraRule**`( )`

  Gets the valid domains for extra rule added by the user. The rule in this case is a chosen domain value for a single variable.

  – **Parameters**

    ∗ `variable` - The variable that has been assigned a domain value by the user.

    ∗ `selectedDomain` - The selected domain value for the variable.

- *GetValidDomainsExtraRule*

  `public` **CLab.ValidDomains GetValidDomainsExtraRule**`( )`

  Gets the valid domains for an extra rule. The rule can be any expression.

  – **Parameters**

    ∗ `expr` - The added expression.

- *InitializeProblemSolver*

`public` **CLab.ValidDomains InitializeProblemSolver**`( )`

Initializes the specified modus, BDD if modus = 0 or CSP if modus = 1.

- **Parameters**
  - `modus` - The modus.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *SetBddCompileMethod*

  `public` **void SetBddCompileMethod**`( )`

  Sets the BDD compile method.

  - **Parameters**
    - `compileMethod` -

- *SetCSPVariableOrdering*

  `public` **void SetCSPVariableOrdering**`( )`

  Sets the CSP variable ordering. ”CspVariableOrdering”

  - **Parameters**
    - `varOrdering` - The variable ordering.

- *SetStatusUpdateCount*

  `assembly` **void SetStatusUpdateCount**`( )`

  Sets the status update count, which is the maximum number of status updates for a specific search. Can be used to set the maximum value of a progress bar.

  - **Parameters**
    - `suc` - The suc.

- *StatusUpdateCount*

  `public` **int StatusUpdateCount**`( )`

  Method which return the maximal number of calls to the status event for a search, depending on which modus is chosen. Can be used to set the max value of a progress bar.

- *UpdateStatus*

  `assembly` **void UpdateStatus**`( )`

  Updates the status of the currently checked value.

  - **Parameters**
    * `value` - The currently checked value.

- *VariableCount*

  `public` **int VariableCount**`( )`

  Returns the number of variables.

CLASS **Common**

Defines common enum types and constant values used within CLab.

DECLARATION

public class Common
: Object

FIELDS

- *BDDINITDBCACHE*

  `public` **int BDDINITDBCACHE**

  The initial cache used by BDD.

- *BDDMAXINCREASE*

  `public` **int BDDMAXINCREASE**

  The max increase of BDD.

- *INITBDDNODES*

  `public` **int INITBDDNODES**

  Number of initial BDD nodes.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  – **Parameters**

    ∗ `obj` -

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

CLASS **ValidDomain**

Class with valid domain information for a particular variable.

DECLARATION

```
public class ValidDomain
        : Object
```

PROPERTIES

- *Domains*

  `public` **System.Collections.Generic.List{System.String} Domains** `{ get; }`

  Gets the valid domains.

- *VarName*

  `public` **string VarName** `{ get; }`

  Gets the name of the variable.

CONSTRUCTORS

- *Constructor*

  `public` **ValidDomain**`( )`

  Initializes a new instance of the ”ValidDomain” class.

  – **Parameters**
    * `varName` - Name of the variable.
    * `typeName` - Name of the type.
    * `typeFormat` - The CPTypes type format.
    * `domains` - The list of domain values.

- *Constructor*

  `public` **ValidDomain**`( )`

  Initializes a new instance of the ”ValidDomain” class.

- **Parameters**
  - `varName` - Name of the variable.
  - `typeName` - Name of the type.
  - `typeFormat` - The CPTypes type format.

METHODS

- *AddValidDomain*

  `assembly` **CLab.ValidDomain AddValidDomain** `( )`

  Adds a valid domain.

  - **Parameters**
    - `domain` - The valid domain to add.

- *EmptyValidDomain*

  `assembly` **void EmptyValidDomain** `( )`

  Empties the valid domain list.

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **ValidDomains**

Class with the valid domains of each variable. This class has a list of ”ValidDomain” objects for each variable. This is the result representation returned out of the library.

DECLARATION

```
public class ValidDomains
      : Object
```

PROPERTIES

- *ValidDoms*

  `public` **System.Collections.Generic.List{CLab.ValidDomain}** **ValidDoms** `{ get; }`

  Gets the list of the valid domains.

CONSTRUCTORS

- *Constructor*

  `public` **ValidDomains** `( )`

  Initializes a new instance of the ”ValidDomains” class.

  – **Parameters**

    ∗ `validDoms` - A list of valid domains.

- *Constructor*

  `public` **ValidDomains** `( )`

  Initializes a new instance of the ”ValidDomains” class.

METHODS

- *addValidDomain*

  `public` **int addValidDomain** `( )`

  Adds the valid domain to the list.

  - **Parameters**
    - `validDomain` - The valid domain to add to the list.

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

6.2 Namespace CLab.BDD

Namespace Contents *Page*

ValidAssignmentData 84

Class which is used to get valid assignment data from the result BDD of the CP problem. Contains data structures which is filled with result data by Buddy_sharp and Buddy.

6.2.1 Classes

CLASS **BDDComparer**

An System.Collections.Generic.IComparer implementation when sorting a list of BDDs, based on their node count. Used in CLab.BDD.Space, when CLab.Auxiliary.Common.CompileMethod is set to cm_ascending.

DECLARATION

public class BDDComparer
: Object

CONSTRUCTORS

- *Constructor*

  `public` **BDDComparer**`( )`

  Initializes a new instance of the BDDComparer" class.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **BDDLayout**

Class representing and making the layout of the BDD problem. The class has mapping from the "CLab.Data.CP" representation to the BDD representation and vice versa.

DECLARATION

```
public class BDDLayout
      : Object
```

PROPERTIES

- *BddVarNum*

  `public` **int BddVarNum** `{ get; }`

  Gets the number of BDD variables.

- *LayoutTypes*

  `public` **System.Collections.Generic.List{CLab.BDD.BDDType} LayoutTypes** `{ get; }`

  Gets the the list of types used by layout.

- *LayoutVariables*

  `public` **System.Collections.Generic.List{CLab.BDD.BDDVariable} LayoutVariables** `{ get; }`

  Gets the variables used by layout.

- *TypeNameToIndex*

  `public` **System.Collections.Hashtable TypeNameToIndex** `{ get; }`

  Gets the hash table mapping type names to index.

- *VariableNameToIndex*

  `public` **System.Collections.Hashtable VariableNameToIndex** `{ get; }`

Gets the hash table mapping variable names to index.

CONSTRUCTORS

- *Constructor*

  `public` **BDDLayout**`( )`

  Initializes a new instance of the ”BDDLayout” class.

  – **Parameters**

    ∗ `cp` - The ”CLab.Data.CP” instance of the problem.

METHODS

- *BooleanVector*

  `public` **System.Collections.Generic.List{System.Int32} BooleanVector**`( )`

  Makes boolean vectors of variables.

  – **Parameters**

    ∗ `nextBDDvar` - The next BDD variable.
    ∗ `varNumber` - The variable number.

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  – **Parameters**

    ∗ `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”CLab.BDD.BDDLayout”.

CLASS **BDDSpace**

Class for making BDDs based on the types, variables and expressions of a CP problem. Uses the ”BDDLayout” mapping of cp variables to BDD variables.

DECLARATION

public class BDDSpace

: Object

PROPERTIES

- *LayoutBdd*

  `public` **buddy_sharp.Bdd LayoutBdd** `{ get; }`

  Gets the BDD representation of the CSP problem before the expressions is compiled in.

- *UnsatisfiableRule*

  `public` **string UnsatisfiableRule** `{ get; }`

  Gets the unsatisfiable rule, if one exist.

CONSTRUCTORS

- *Constructor*

  `public` **BDDSpace**`( )`

  Initializes a new instance of the ”BDDSpace” class.

  - **Parameters**
    * `cp` - The ”CP” instance of the problem.
    * `layout` - The ”BDDLayout” instance of the problem.
    * `symbols` - The ”CLab.Data.Symbols” instance of the problem.
    * `clabBdd` - The ”ClabBDD” instance of the problem.

METHODS

- *Abs*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} Abs**`( )`

  Calculates the absolute value of a 2-complementary BDD representation.

  - **Parameters**
    - `x` - The 2-complementary BDD representation.

- *Add*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} Add**`( )`

  Calculates the result of the Add operation between two 2-complementary BDD representations.

  - **Parameters**
    - `x` - A 2-complementary BDD representation.
    - `y` - Another 2-complementary BDD representatino.

- *Bool2Integer*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} Bool2Integer**`( )`

  Converts the specified BDD ”x” to a 2-complementary BDD representation.

  - **Parameters**
    - `x` - The BDD.

- *Compile*

  `public` **buddy_sharp.Bdd Compile**`( )`

  Compiles the specified expression.

  - **Parameters**
    - `expression` - The expression.

- *CompileRules*

  `public` **buddy_sharp.Bdd CompileRules** `( )`

  Tries to compile all rules defined in the problem file. Each expression is compiled to a BDD. Afterwards the BDDs are and'ed together, depending on which ordering is chosen. The order of the compilation can be set three different ways: Static: In the same order the expressions are written in the problem. Ascending: The order depends on the node count of each expression. Smallest node count is and'ed with the result BDD first. Dynamic: The two smallest BDDs are and'ed, and the result is placed back in the queue. This is continued until we have only one element left in the queue, which is the final result BDD.

  - **Parameters**
    - `method` - The "CLab.Auxiliary.Common.CompileMethod" representing which ordering to use.

- *Div*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} Div** `( )`

  Calculates the result of the division operation between two 2-complementary BDD representations.

  - **Parameters**
    - `x` - 2-complementary BDD representation of the numerator.
    - `y` - 2-complementary BDD representation of the denominator.

- *Equal*

  `public` **buddy_sharp.Bdd Equal** `( )`

  Calculates a BDD representing the equal operation on two 2-complementary BDD representations.

  - **Parameters**
    - `x` - The left 2-complementary BDD representation.
    - `y` - The right 2-complementary BDD representation.

- *Equals*

`public` **bool Equals**`( )`

Determines whether the specified is equal to the current .

- **Parameters**
  * `obj` -

- *Expr2BddVec*

  `public` **CLab.BDD.Bval Expr2BddVec**`( )`

  Compiles a expression into a ”Bval” object.

  - **Parameters**
    * `expression` - The expression to be compiled.

- *Extend*

  `public` **void Extend**`( )`

  Extends the specified ”x” with ”k” elements.

  - **Parameters**
    * `x` - The 2-complementary BDD representation to be extended.
    * `k` - Number of elements to extend with.

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

`public` **System.Type GetType** `( )`

Gets the of the current instance.

- *Integer2Bool*

  `public` **buddy_sharp.Bdd Integer2Bool** `( )`

  Converts the specified 2-complementary BDD represantation ”x” to a BDD.

  – **Parameters**

    ∗ `x` - The 2-complementary representation.

- *LessThan*

  `public` **buddy_sharp.Bdd LessThan** `( )`

  Calculates a BDD representing the less-than operation on two 2-complementary BDD representations.

  – **Parameters**

    ∗ `x` - The left 2-complementary BDD representation.
    ∗ `y` - The right 2-complementary BDD representation.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *MkSameSize*

  `public` **void MkSameSize** `( )`

  Extends the smallest of the specified ”x” or ”y”, so the lists gets equal length. Modifies one of them, if they have unequal lenght.

  – **Parameters**

    ∗ `x` - 2-complementary representation.
    ∗ `y` - 2-complementary representation.

- *MkVal*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} MkVal** `( )`

  Makes a 2-complementary BDD representation of an integer value.

  - **Parameters**
    - `intVal` - The integer value.

- *MkVar*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} MkVar** `( )`

  Makes a 2-complementary BDD representation of the boolan vector representing the bdd variable.

  - **Parameters**
    - `Bddvar` - The boolean vector representing the bdd variable.

- *Mod*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} Mod** `( )`

  Calculates the result of the modulo operation between two 2-complementary BDD representations.

  - **Parameters**
    - `x` - 2-complementary BDD representation of the numerator.
    - `y` - 2-complementary BDD representation of the denominator.

- *Mul*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} Mul** `( )`

  Calculates the result of the multiplication operation between two 2-complementary BDD representations.

  - **Parameters**
    - `x` - A 2-complementary BDD representation.
    - `y` - Another 2-complementary BDD representation.

- *Neg*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} Neg** `( )`

  Negates the specified 2-complentary BDD representation ”x”.

  - **Parameters**
    - `x` - 2-complementary BDD representation to be negated

- *ShiftLeft*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} ShiftLeft** `( )`

  Shifts each bit ”shiftAmount” positions to the left.

  - **Parameters**
    - `x` - The 2-complementary representation of a BDD.
    - `shiftAmount` - Number of positions ”x” should be shifted left.

- *Truncate*

  `public` **void Truncate** `( )`

  Truncates the specified x.

  - **Parameters**
    - `x` - The x.

CLASS **BDDType**

Abstract class representing variable types. Used by ”CLab.BDD.Layout”.

DECLARATION

```
public class BDDType
    : Type
```

PROPERTIES

- *BDDvarNum*

  `public` **int BDDvarNum** `{ get;  set; }`

  Gets or sets the BDD variable number.

- *DomainSize*

  `public` **int DomainSize** `{ get;  set; }`

  Gets or sets the size of the domain.

- *TypeName*

  `public` **string TypeName** `{ get;  set; }`

  Gets or sets the string representation of the name of the type.

CONSTRUCTORS

- *Constructor*

  `public` **BDDType**`( )`

  Initializes a new instance of the ”BDDLayoutType” class.

  – **Parameters**

    ∗ `layoutTypeName` - Name of the layout type.
    ∗ `bddVarNum` - The number of BDD variables.
    ∗ `domainSize` - Size of the domain.

METHODS

- *BooleanVarNumber*

  `public` **int BooleanVarNumber**`( )`

  Calculates the number of BDD variables needed, using the domain size.

  - **Parameters**
    - `domainSize` - Size of the domain.

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetDomainValue*

  `public` **string GetDomainValue**`( )`

  Method that gets the Domain value at ”index”. Index can be ”0” to ”domain size - 1”.

  - **Parameters**
    - `index` - The index of the domain value.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”CLab.BDD.BDDLayoutType”.

- *TypeFormat*

  `public` **CLab.CPTypes TypeFormat**`( )`

  Returns a constant from the ”CPTypes” enumeration, representing the current type.

CLASS **BDDTypeBool**

Class representing the boolean type.

DECLARATION

```
public class BDDTypeBool
      : BDDType
```

PROPERTIES

- *BDDvarNum*

  `public` **int BDDvarNum** `{ get;  set; }`

  Gets or sets the BDD variable number.

- *DomainSize*

  `public` **int DomainSize** `{ get;  set; }`

  Gets or sets the size of the domain.

- *TypeName*

  `public` **string TypeName** `{ get;  set; }`

  Gets or sets the string representation of the name of the type.

CONSTRUCTORS

- *Constructor*

  `public` **BDDTypeBool**`( )`

  Initializes a new instance of the ”BDDLayoutTypeBool” class.

  – **Parameters**

    ∗ `typeName` - Name of the type.
    ∗ `domSize` - Size of the domain.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetDomainValue*

  `public` **string GetDomainValue**`( )`

  Method that gets the Domain value at ”index”. Index can be ”0” to ”domain size - 1”.

  - **Parameters**
    - `index` - The index of the domain value.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”CLab.BDD.BDDLayoutTypeBool”.

- *TypeFormat*

  `public` **CLab.CPTypes TypeFormat**`( )`

  Returns the ”CPTypes.bool_type” constant.

CLASS **BDDTypeEnum**

Class representing the enumeration type.

DECLARATION

```
public class BDDTypeEnum
    : BDDType
```

PROPERTIES

- *BDDvarNum*

  `public` **int BDDvarNum** `{ get;  set; }`

  Gets or sets the BDD variable number.

- *DomainSize*

  `public` **int DomainSize** `{ get;  set; }`

  Gets or sets the size of the domain.

- *IndexToValue*

  `public` **System.Collections.ArrayList IndexToValue** `{ get; }`

  Gets the list mapping index to value.

- *TypeName*

  `public` **string TypeName** `{ get;  set; }`

  Gets or sets the string representation of the name of the type.

- *ValueToIndex*

  `public` **System.Collections.Hashtable ValueToIndex** `{ get; }`

  Gets the hashtable mapping value to index.

CONSTRUCTORS

- *Constructor*

  `public` **BDDTypeEnum** `( )`

  Initializes a new instance of the ”BDDLayoutTypeEnum” class.

  – **Parameters**

  ∗ `typeName` - Name of the type.
  ∗ `domainSize` - Size of the domain.
  ∗ `indexToValue` - List mapping index to value.
  ∗ `valueToIndex` - Hashtable mapping value to index.

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  – **Parameters**

  ∗ `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetDomainValue*

  `public` **string GetDomainValue** `( )`

  Method that gets the Domain value at ”index”. Index can be ”0” to ”domain size - 1”.

  – **Parameters**

  ∗ `index` - The index of the domain value.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString** `( )`

  Returns a ”System.String” that represents the current ”CLab.BDD.BDDLayoutTypeEnum”.

- *TypeFormat*

  `public` **CLab.CPTypes TypeFormat** `( )`

  Returns the ”CPTypes.enum_type” constant.

CLASS **BDDTypeRange**

Class representing the range type.

DECLARATION

```
public class BDDTypeRange
    : BDDType
```

PROPERTIES

- *BDDvarNum*

  `public` **int BDDvarNum** `{ get;  set; }`

  Gets or sets the BDD variable number.

- *DomainSize*

  `public` **int DomainSize** `{ get;  set; }`

  Gets or sets the size of the domain.

- *StartOfRange*

  `public` **int StartOfRange** `{ get; }`

  Gets the start of the range.

- *TypeName*

  `public` **string TypeName** `{ get;  set; }`

  Gets or sets the string representation of the name of the type.

CONSTRUCTORS

- *Constructor*

  `public` **BDDTypeRange**`( )`

  Initializes a new instance of the ”BDDLayoutTypeRange” class.

- **Parameters**
  - `typeName` - Name of the type.
  - `startOfRange` - The start of the range.
  - `domainSize` - Size of the domain.

Methods

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetDomainValue*

  `public` **string GetDomainValue**`( )`

  Method that gets the Domain value at ”index”. Index can be ”0” to ”domain size - 1”.

  - **Parameters**
    - `index` - The index of the domain value.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”CLab.BDD.BDDLayoutRange”.

- *TypeFormat*

  `public` **CLab.CPTypes TypeFormat**`( )`

  Returns the ”CPTypes.range_type” constant..

CLASS **BDDVariable**

Class representing a CLab variable.

DECLARATION

```
public class BDDVariable
      : Object
```

PROPERTIES

- *BddVar*

  `public` **System.Collections.Generic.List{System.Int32} BddVar** `{ get; }`

  Gets the BDD variable list.

- *TypeIndex*

  `public` **int TypeIndex** `{ get; }`

  Gets the index of the belonging type.

CONSTRUCTORS

- *Constructor*

  `public` **BDDVariable**`( )`

  Initializes a new instance of the ”BDDLayoutVariable” class.

  – **Parameters**
    * `bddVar` - The BDD variables, which together makes a Clab variable.
    * `typeIndex` - Index of belonging type

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

- **Parameters**
  - `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString** `( )`

  Returns a ”System.String” that represents the current ”CLab.Data.BDDLayoutVariable”.

CLASS **Bval**

Class for BDD representation of the configuration space of a CP expression.

DECLARATION

```
public class Bval
    : Object
```

PROPERTIES

- *BDDList*

  `public` **System.Collections.Generic.List{buddy_sharp.Bdd} BDDList** `{ get; set; }`

  Gets or sets the BDD list.

- *DefCon*

  `public` **buddy_sharp.Bdd DefCon** `{ get; set; }`

  Gets or sets the default configuration.

CONSTRUCTORS

- *Constructor*

  `public` **Bval** `( )`

  Initializes a new instance of the ”Bval” class.

  – **Parameters**

    * `bddList` - The list of BDDs
    * `defcon` - The default configuration.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **ClabBDD**

The main class for finding valid solutions the BDD way. The class is designed to be used by ”CLab.Clab”, which provides the data needed for this class to work.

DECLARATION

```
public class ClabBDD
      : Object
```

PROPERTIES

- *Layout*

  `public` **CLab.BDD.BDDLayout Layout** `{ get; }`

  Gets the ”BDDLayout” of the problem.

- *Space*

  `public` **CLab.BDD.BDDSpace Space** `{ get; }`

  Gets the ”BDDSpace” of the problem.

- *UnsatisfiableRule*

  `public` **string UnsatisfiableRule** `{ get; }`

  Gets the unsatisfiable rule if one exist.

CONSTRUCTORS

- *Constructor*

  `public` **ClabBDD**`( )`

  Initializes a new instance of the ”ClabBDD” class.

  – **Parameters**

    ∗ `clab` - The current Clab instance.
    ∗ `cp` - The cp object of the problem.

  * `symbols` - The symbols object of the problem.
  * `initdbcache` - The initial db cache.
  * `initbddnodes` - The initial number of bdd nodes.
  * `maxincrease` - The maximum increase number.

METHODS

- *CompileAllExpressions*

  `public` **buddy_sharp.Bdd CompileAllExpressions**`( )`

  Compiles all expressions.

  - **Parameters**
    * `compileMethod` - The compile method.

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

`public` **System.Type GetType**`( )`

Gets the of the current instance.

- *GetValidDomains*

  `public` **CLab.ValidDomains GetValidDomains**`( )`

  Gets the valid domains.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *printout*

  `public` **void printout**`( )`

  Prints the specified list of BDDs.

  - **Parameters**
    - `b` - The list of BDDs.

- *SetCompileMethod*

  `public` **void SetCompileMethod**`( )`

  Sets the compile method.

  - **Parameters**
    - `compileMethod` - The compile method.

- *UpdateResultBddExpr*

  `public` **buddy_sharp.Bdd UpdateResultBddExpr**`( )`

  Updates the result BDD with the results from an extre expression.

  - **Parameters**
    - `extraExpr` - The extra expression.

- *UpdateResultBDDUserChoice*

  `public` **buddy_sharp.Bdd UpdateResultBDDUserChoice**`( )`

  Updates the result BDD with a user chosen variable and domain value. Makes an expression out of the variable and value, and ands it with the old result BDD.

  - **Parameters**
    * `var` - The chosen variable.
    * `domain` - The chosen domain value.

- *UpdateStatus*

  `public` **void UpdateStatus**`( )`

  Updates the CLab status method.

  - **Parameters**
    * `value` - The value.

CLASS **PQbdd**

Priority queue node with IComparable implementation

DECLARATION

```
public class PQbdd
    : Object
```

PROPERTIES

- *BDD*

  `public` **buddy_sharp.Bdd BDD** `{ get; }`

  Gets the BDD.

- *Size*

  `public` **int Size** `{ get; }`

  Gets the size.

CONSTRUCTORS

- *Constructor*

  `public` **PQbdd**`( )`

  Initializes a new instance of the ”PQbdd” class.

  - **Parameters**
    - `bc` - The BDD.

METHODS

- *CompareTo*

  `public` **int CompareTo**`( )`

  IComparable implementation. Compares PQbdds based on their size

  - **Parameters**
    - `obj` - PQbdd object

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **ValidAssignmentData**

Class which is used to get valid assignment data from the result BDD of the CP problem. Contains data structures which is filled with result data by Buddy_sharp and Buddy.

DECLARATION

public class ValidAssignmentData
    : Object

PROPERTIES

- *BddVarTocpVar*

  `public` **System.Int32[] BddVarTocpVar** `{ get; }`

  Gets the Bdd variable to CP variable mapping.

- *CpVarNum*

  `public` **int CpVarNum** `{ get; }`

  Gets the number of CP variables.

- *Dom*

  `public` **System.Int32[] Dom** `{ get; }`

  Gets the list of domain size for each variable.

- *DomStart*

  `public` **System.Int32[] DomStart** `{ get; }`

  Gets the list with the index of the BDD variable where the encoding of each CP variable starts. DomStart[i] = index of the BDD variable where the encoding of CSP variable ”i” starts

CONSTRUCTORS

- *Constructor*

  `public` **ValidAssignmentData** `( )`

  Initializes a new instance of the ”ValidAssignmentData” class.

  - **Parameters**
    - `layout` - The ”BDDLayout” instance.

METHODS

- *ClabEnd*

  `public` **void** **ClabEnd** `( )`

  Runs the ClabEnd method of ”buddy_sharp.Bdd”.

- *CreateValidAssignmentData*

  `public` **void** **CreateValidAssignmentData** `( )`

  Method to create the valid assignment data. Use the Bdd.valExist(i, j) method to retrieve the results.

  - **Parameters**
    - `resultBdd` - The result BDD to create the valid assigment data on.

- *Equals*

  `public` **bool** **Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void** **Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”ValidAssignmentData”.

6.3 Namespace CLab.CSP

Class representing and making the layout of the CSP problem. The class has mapping from the CLab ”CLab.Data.CP” representation to the CSP representation and vice versa.

CSPType 101

An abstract class used by classes representing types in the CSP part of CLab. The different possible ”values”, like strings for the enumeration type, or integers for the range type, are all encoded as ranges. All of the types must have an end point.

CSPTypeBool 104

A class for representing boolean types in the CSP part of CLab. The different possible domain values for a range are encoded as a range with an end point. The start point is assumed always to be 0. Implements the abstract CSPType class.

CSPTypeEnum 107

A class for representing enumeration types in the CSP part of CLab. The different possible domain values for an enumeration are encoded as a range with an end point. The start point is assumed always to be 0. Implements the abstract CSPType class.

CSPTypeRange 110

A class for representing range types in the CSP part of CLab. The different possible domain values for a range are encoded as a range with a start point and an end point. Implements the abstract CSPType class.

6.3.1 Classes

CLASS **ClabCSP**

The main class for finding valid solutions with the CSP approach. The class is designed to be used by ”CLab.Clab”, which provides the data needed for this class to work.

DECLARATION

public class ClabCSP
: Object

PROPERTIES

- *Csp*

  `public` **Casper.Csp Csp** `{ get; }`

  Gets the ”Casper.CSP” object of the problem.

- *Layout*

  `public` **CLab.CSP.CSPLayout Layout** `{ get; }`

  Gets the ”CSPLayout” of the problem.

CONSTRUCTORS

- *Constructor*

  `public` **ClabCSP**`( )`

  Initializes a new instance of the ”ClabCSP” class.

  – **Parameters**
    * `clab` - The ”CLab.Clab” object.
    * `cp` - The ”CLab.Data.CP” object of the problem.
    * `symbols` - The ”CLab.Data.Symbols” object of the problem.

METHODS

- *AddExtraRule*

  `public` **void AddExtraRule**`( )`

  Adds an extra rule to the problem. Does not run a search for valid solutions, this has to be done, to get the reduced valid domaines after adding the new expression.

  - **Parameters**
    - `newExpr` - The new expression.

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *GetValidDomains*

  `public` **CLab.ValidDomains GetValidDomains** `( )`

  Updates the ”CLab.ValidDomains” instance with the domain information from the current ”Casper.CSP” object. Before a search is run, all domain values are possible and returned.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *ObserveStatusChanged*

  `public` **void ObserveStatusChanged** `( )`

  Observes the status for changes, and updates the current ”CLab.Clab” instance.

  – **Parameters**
    ∗ `currentlyRunVar` - The currently run var.

- *RunCSPSearch*

  `public` **void RunCSPSearch** `( )`

  Runs the CSP search with the expressions defined, and return the valid domaines for the problem.

- *RunCSPSearchUserChoice*

  `public` **void RunCSPSearchUserChoice** `( )`

  Runs the CSP search reduced with a user chosen domain value for a variable. Returns the valid domaines for the problem after this reduction.

  – **Parameters**
    ∗ `var` - The chosen variable.
    ∗ `domVal` - The chosen domain value.

- *SetVariableOrdering*

  `public` **void SetVariableOrdering** `( )`

  Sets the variable ordering.

  - **Parameters**
    - `variableOrder` - The variable ordering.

CLASS **CSPExpressionBuilder**

Class for making Casper's CSP representation of the expressions. "Casper.Data.CSPExpr".

DECLARATION

```
public class CSPExpressionBuilder
    : Object
```

PROPERTIES

- *ConstraintGraph*

  `assembly` **Casper.Data.ConstraintGraph ConstraintGraph** `{ get; }`

  Gets the constraint graph.

- *CSPExpressions*

  `public` **Casper.Data.CSPExpressions CSPExpressions** `{ get; }`

  Gets the CSP expressions.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExpressionBuilder**`( )`

  Initializes a new instance of the "CSPExpressionBuilder" class.

  – **Parameters**
    * `layout` - The "Casper.CSP" layout.
    * `cp` - The "CLab.Data.CP" instance of this problem.
    * `symbols` - The "CLab.Data.Symbols" instance for this problem.

METHODS

- *BuildCSPExpr*

  `public` **Casper.Data.CSPExpr BuildCSPExpr** `( )`

  Builds the CSP expr for an ”CLab.Data.ExpressionId” expression.

  - **Parameters**
    * `idexpr` - The ”CLab.Data.ExpressionId” expression.

- *BuildCSPExpr*

  `public` **Casper.Data.CSPExpr BuildCSPExpr** `( )`

  Builds the CSP expr for an ”CLab.Data.ExpressionInt” expression.

  - **Parameters**
    * `iExpr` - The ”CLab.Data.ExpressionInt” expression.

- *BuildCSPExpr*

  `public` **Casper.Data.CSPExpr BuildCSPExpr** `( )`

  Builds the CSP expr for an ”CLab.Data.ExpressionNotNeg” expression.

  - **Parameters**
    * `nnExpr` - The ”CLab.Data.ExpressionNotNeg” expression.

- *BuildCSPExpr*

  `public` **Casper.Data.CSPExpr BuildCSPExpr** `( )`

  Builds the CSP expr for an ”CLab.Data.ExpressionBinary” expression.

  - **Parameters**
    * `bExpr` - The ”CLab.Data.ExpressionBinary” expression.

- *BuildCSPExpression*

  `public` **Casper.Data.CSPExprWrapper BuildCSPExpression** `( )`

  Builds a csp expression out of the passed in clab expression. Updates the constraint graph with new data from the current expression.

- **Parameters**
  - `expr` - The clab expression.

- *BuildCSPExpressions*

  `public` **Casper.Data.CSPExpressions BuildCSPExpressions** `( )`

  Builds csp expressions of all clab expressions defined in the "CLab.Data.CP" instance. Also updates the Constraint graph of the problem.

- *ConvertOperator*

  `public` **Casper.StaticData.Operators ConvertOperator** `( )`

  Converts a clab operator to a Casper operator. "Casper.StaticData.Operators".

  - **Parameters**
    - `input` - The input.

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **CSPLayout**

Class representing and making the layout of the CSP problem. The class has mapping from the CLab ”CLab.Data.CP” representation to the CSP representation and vice versa.

DECLARATION

```
public class CSPLayout
    : Object
```

PROPERTIES

- *CpVarToVarID*

  `public` **System.Collections.Hashtable CpVarToVarID** `{ get; }`

  Gets the ”CLab.Data.Variable.VariableName” property to the ”Casper.Data.CasperVarDom.VarID” property mapping.

- *TypeNameToIndex*

  `public` **System.Collections.Hashtable TypeNameToIndex** `{ get; }`

  Gets the ”CLab.Data.Variable.TypeName” property to the index of this type mapping.

- *Types*

  `public` **System.Collections.Generic.List{CLab.CSP.CSPType} Types** `{ get; }`

  Gets the list of CSP types.

- *VarIDToCPVar*

  `public` **System.Collections.Generic.List{System.String} VarIDToCPVar** `{ get; }`

  Gets the ”Casper.Data.CasperVarDom.VarID” property to the ”CLab.Data.Variable.VariableName” property mapping.

- *VarIDToTypeName*

  `public` **System.Collections.Generic.List**{**System.String**} **VarIDToTypeName** `{ get; }`

  Gets the ”Casper.Data.CasperVarDom.VarID” property to the ”CLab.Data.Variable.TypeName” property mapping.

CONSTRUCTORS

- *Constructor*

  `public` **CSPLayout**`( )`

  Initializes a new instance of the ”CSPLayout” class.

  - **Parameters**
    - `cp` - The ”CLab.Data.CP” instance of the problem.

METHODS

- *Equals*

  `public` **bool** **Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void** **Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetCasperVarDomIndex*

  `public` **int** **GetCasperVarDomIndex**`( )`

  Gets the index of the ”Casper.Data.CasperVarDom” object with the name ”cpVarName”.

- **Parameters**
  - `cpVarName` - Name of the Clab variable.

- *GetCPVarFromVarID*

  `public` **string GetCPVarFromVarID**`( )`

  Gets the variable name, (CLab.Data.Variable.VariableName"), from the passed inn variable ID, (Casper.Data.CasperVarDom.VarID").

  - **Parameters**
    - `varID` - The var ID.

- *GetDomainIntFromDomainString*

  `public` **int GetDomainIntFromDomainString**`( )`

  Gets the domain value's int representation.

  - **Parameters**
    - `cpVarName` - Name of the CLab variable.
    - `domainString` - The domain string value.

- *GetDomainStringFromDomainInt*

  `public` **string GetDomainStringFromDomainInt**`( )`

  Gets the domain value's string representation.

  - **Parameters**
    - `cpVarName` - Name of the CLab variable.
    - `domainInt` - The domain int value.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *GetTypeFormat*

  `public` **CLab.CPTypes GetTypeFormat** `( )`

  Gets the type format. ”CPTypes”.

  – **Parameters**
    * `typeName` - Name of the type.

- *GetTypeName*

  `public` **string GetTypeName** `( )`

  Gets the name of the variable’s type.

  – **Parameters**
    * `varName` - The CLab variable name.

- *GetVarIDFromCPVar*

  `public` **int GetVarIDFromCPVar** `( )`

  Gets the variable ID, (Casper.Data.CasperVarDom.VarID”), from the passed in variable name, (CLab.Data.Variable.VariableName”).

  – **Parameters**
    * `cpVarName` - Name of the cp var.

- *MakeCasperVarDoms*

  `public` **System.Collections.Generic.List{Casper.Data.CasperVarDom} MakeCasperVarDoms** `( )`

  Makes the casper variables with domain values, ”Casper.Data.CasperVarDom”, and the mapping from ”CLab.Data.CP” representation, to this representation. The method is for internal use of this class.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”CLab.CSP.CSPLayout”.

CLASS **CSPType**

An abstract class used by classes representing types in the CSP part of CLab. The different possible ”values”, like strings for the enumeration type, or integers for the range type, are all encoded as ranges. All of the types must have an end point.

DECLARATION

```
public class CSPType
      : Object
```

PROPERTIES

- *End*

  `public` **int End** `{ get;  set; }`

  Gets or sets the end of the range.

- *TypeName*

  `public` **string TypeName** `{ get;  set; }`

  Gets or sets the name of the type.

CONSTRUCTORS

- *Constructor*

  `public` **CSPType**`( )`

  Initializes a new instance of the ”CSPType” class.

  – **Parameters**

    * `typeName` - Name of the type.
    * `end` - The end value.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetIntRepOfDomainString*

  `public` **int GetIntRepOfDomainString**`( )`

  Gets the int representation of the string representation of a domain value. Has to be implemented by classes implmementing this abstract class.

  - **Parameters**
    - `domainString` - The string representation of a domain value.

- *GetStringRepOfDomainInt*

  `public` **string GetStringRepOfDomainInt**`( )`

  Gets the string representation of an int within the range of a type. Has to be implemented by classes implmementing this abstract class.

  - **Parameters**

  * `domainInt` - An integer representing a domain value.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”CLab.CSP.CSPType”.

- *TypeFormat*

  `public` **CLab.CPTypes TypeFormat**`( )`

  Returns a constant from the ”CPTypes” enumeration, representing the current type.

CLASS **CSPTypeBool**

A class for representing boolean types in the CSP part of CLab. The different possible domain values for a range are encoded as a range with an end point. The start point is assumed always to be 0. Implements the abstract CSPType class.

DECLARATION

```
public class CSPTypeBool
      : CSPType
```

PROPERTIES

- *End*

  `public` **int End** `{ get;  set; }`

  Gets or sets the end of the range.

- *TypeName*

  `public` **string TypeName** `{ get;  set; }`

  Gets or sets the name of the type.

CONSTRUCTORS

- *Constructor*

  `public` **CSPTypeBool**`( )`

  Initializes a new instance of the ”CSPTypeBool” class.

  – **Parameters**

    ∗ `typeName` - Name of the type.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetIntRepOfDomainString*

  `public` **int GetIntRepOfDomainString**`( )`

  Gets the int representation of the string representation of a domain value.

  - **Parameters**
    - `domainString` - The string representation of a domain value.

- *GetStringRepOfDomainInt*

  `public` **string GetStringRepOfDomainInt**`( )`

  Gets the string representation of an int within the range of a type.

  - **Parameters**
    - `domainInt` - An integer representing a domain value.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”CLab.CSP.CSPTypeBool”.

- *TypeFormat*

  `public` **CLab.CPTypes TypeFormat**`( )`

  Returns the ”CPTypes.bool_type” constant.

CLASS **CSPTypeEnum**

A class for representing enumeration types in the CSP part of CLab. The different possible domain values for an enumeration are encoded as a range with an end point. The start point is assumed always to be 0. Implements the abstract CSPType class.

DECLARATION

public class CSPTypeEnum

: CSPType

PROPERTIES

- *End*

  `public` **int End** `{ get;  set; }`

  Gets or sets the end of the range.

- *TypeName*

  `public` **string TypeName** `{ get;  set; }`

  Gets or sets the name of the type.

CONSTRUCTORS

- *Constructor*

  `public` **CSPTypeEnum**`( )`

  Initializes a new instance of the ”CSPTypeEnumerator” class.

  – **Parameters**

    ∗ `typeName` - Name of the type.

METHODS

- *AddConstant*

  `public` **int AddConstant** `( )`

  Adds a new enumeration domain value (constant). This method makes a mapping from string representation to int representation and vice versa.

  - **Parameters**
    - `constant` - The constant.

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetIntRepOfDomainString*

  `public` **int GetIntRepOfDomainString** `( )`

  Gets the int representation of the string representation of a domain value. Uses the internal mapping from string to int.

  - **Parameters**
    - `domainString` - The string representation of a domain value.

- *GetStringRepOfDomainInt*

  `public` **string GetStringRepOfDomainInt** `( )`

  Gets the string representation of an int within the range of a type. Uses the internal mapping from int to string.

  - **Parameters**
    - `domainInt` - An integer representing an enumeration domain value (constant).

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString** `( )`

  Returns a ”System.String” that represents the current ”CLab.CSP.CSPTypeEnum”.

- *TypeFormat*

  `public` **CLab.CPTypes TypeFormat** `( )`

  Returns the ”CPTypes.enum_type” constant.

CLASS **CSPTypeRange**

A class for representing range types in the CSP part of CLab. The different possible domain values for a range are encoded as a range with a start point and an end point. Implements the abstract CSPType class.

DECLARATION

```
public class CSPTypeRange
    : CSPType
```

PROPERTIES

- *End*

  `public` **int End** `{ get;  set; }`

  Gets or sets the end of the range.

- *Start*

  `public` **int Start** `{ get; }`

  Gets the start point of the range.

- *TypeName*

  `public` **string TypeName** `{ get;  set; }`

  Gets or sets the name of the type.

CONSTRUCTORS

- *Constructor*

  `public` **CSPTypeRange**`( )`

  Initializes a new instance of the ”CSPTypeRange” class.

  – **Parameters**

    * `typeName` - Name of the type.

  * `start` - The start of the range.
  * `end` - The end of the range.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetIntRepOfDomainString*

  `public` **int GetIntRepOfDomainString**`( )`

  Gets the int representation of the string representation of a domain value.

  - **Parameters**
    * `domainString` - The string representation of a domain value.

- *GetStringRepOfDomainInt*

  `public` **string GetStringRepOfDomainInt**`( )`

  Gets the string representation of an int within the range.

- **Parameters**
  - `domainInt` - The domain int.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”CLab.CSP.CSPTypeRange”.

- *TypeFormat*

  `public` **CLab.CPTypes TypeFormat**`( )`

  Returns the ”CPTypes.range_type” constant..

6.4 Namespace CLab.Data

Namespace Contents *Page*

6.4.1 Classes

CLASS **CP**

The CP object contains the internal data representation, parsed from the input XML file

DECLARATION

```
public class CP
    : Object
```

PROPERTIES

- *Author*

  public **string** **Author** { get; set; }

  Optional header information in the XML file. Author of the XML data (optional metadata)

- *Date*

  public **string** **Date** { get; set; }

  Optional header information in the XML file. Timestamp of the XML data (optional metadata)

- *Description*

  public **string** **Description** { get; set; }

  Optional header information in the XML file. Description of the XML data (optional metadata)

- *Rules*

  public **System.Collections.Generic.List{CLab.Data.Expression}** **Rules** { get; }

  Gets the list of rule (expression) objects in the internal data representation

- *Types*

  `public` **System.Collections.Generic.List{CLab.Data.Type}** **Types** `{ get; }`

  Gets the list of type objects in the internal data representation

- *Variables*

  `public` **System.Collections.Generic.List{CLab.Data.Variable}** **Variables** `{ get; }`

  Gets the List of variable objects in the internal data representation.

CONSTRUCTORS

- *Constructor*

  `public` **CP**`( )`

  Initializes a new instance of the ”CP” class.

  – **Parameters**
    * `xmlFilename` - The XML filename.

- *Constructor*

  `public` **CP**`( )`

  Initializes a new instance of the ”CP” class.

  – **Parameters**
    * `stream` - The stream.

METHODS

- *AddRule*

  `public` **void** **AddRule**`( )`

  Adds a rule (expression) object to the internal data representation

  – **Parameters**

    * `newRule` - The rule to add

- *AddType*

  `public` **void AddType**`( )`

  Adds a type object to the internal data representation

  - **Parameters**
    * `newType` - The type object to add

- *AddVariable*

  `public` **void AddVariable**`( )`

  Adds a variable object to the internal data representation

  - **Parameters**
    * `newVariable` - The variable object to add

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **Expression**

Abstract class for the different types of Expressions

DECLARATION

public class Expression
  : Object

PROPERTIES

- *Type*

  `public` **CLab.Common.ExprType Type** `{ get; }`

  Returns the type of this expression

CONSTRUCTORS

- *Constructor*

  `public` **Expression** `( )`

  Constructor

  - **Parameters**
    * `type` - The type of the expression: This can be either an operator working on a left and right side, or an id or integer expression

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *RunCSPBuildExpr*

  `assembly` **Casper.Data.CSPExpr RunCSPBuildExpr**`( )`

  Abstract method which runs CSPBuildExpr in CLab.CSP.CSPExpressionBuilder Double dispatch pattern

  – **Parameters**

    * `builder` - The CSPExpressionBuilder.

- *Type2oper*

  `public` **string Type2oper**`( )`

  Converts the type to an operator sign for printout

  – **Parameters**

* `type` - The operator to print out

CLASS **ExpressionBinary**

An expression consisting of a left and right side and an operator. Implements the abstract Expression class.

DECLARATION

```
public class ExpressionBinary
    : Expression
```

PROPERTIES

- *Left*

  `assembly` **CLab.Data.Expression Left** { `get;` }

  Returns left side expression.

- *Right*

  `assembly` **CLab.Data.Expression Right** { `get;` }

  Returns right side expression

- *Type*

  `public` **CLab.Common.ExprType Type** { `get;` }

  Returns the type of this expression

CONSTRUCTORS

- *Constructor*

  `public` **ExpressionBinary**`( )`

  Initializes a new instance of the ”ExpressionBinary” class.

  – **Parameters**
    * `left` - The left expression.
    * `type` - The operator.
    * `right` - The right expression.

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *RunCSPBuildExpr*

  `assembly` **Casper.Data.CSPExpr RunCSPBuildExpr** `( )`

  Abstract method which runs CSPBuildExpr in CLab.CSP.CSPExpressionBuilder Double dispatch pattern

  - **Parameters**
    - `builder` - The CSPExpressionBuilder.

- *ToString*

  `public` **string ToString**`( )`

  Returns a textual description of the expression

- *Type2oper*

  `public` **string Type2oper**`( )`

  Converts the type to an operator sign for printout

  - **Parameters**
    - `type` -

CLASS **ExpressionId**

An expression consisting of a single id-element. Implements the abstract expression class.

DECLARATION

```
public class ExpressionId
    : Expression
```

PROPERTIES

- *Id*

  `public` **string Id** `{ get; }`

  Returns the Id value

- *Type*

  `public` **CLab.Common.ExprType Type** `{ get; }`

  Returns the type of this expression

CONSTRUCTORS

- *Constructor*

  `public` **ExpressionId** `( )`

  Initializes a new instance of the ”ExpressionId” class.

  – **Parameters**
    * `id` - A string with the name of the variable or an enumeration constant

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *RunCSPBuildExpr*

  `assembly` **Casper.Data.CSPExpr RunCSPBuildExpr**`( )`

  Abstract method which runs CSPBuildExpr in CLab.CSP.CSPExpressionBuilder. Double dispatch pattern.

  - **Parameters**
    * `builder` - The CSPExpressionBuilder.

- *ToString*

  `public` **string ToString**`( )`

Returns a ”System.String” that represents the current ”System.Object”.

- *Type2oper*

  `public` **string Type2oper**`( )`

  Converts the type to an operator sign for printout

  - **Parameters**
    - `type` -

CLASS **ExpressionInt**

Expression consisting of an integer. Implements the abstract Expression class.

DECLARATION

```
public class ExpressionInt
      : Expression
```

PROPERTIES

- *IntegerValue*

  `public` **int IntegerValue** `{ get; }`

  Returns the integer value of this expression.

- *Type*

  `public` **CLab.Common.ExprType Type** `{ get; }`

  Returns the type of this expression

CONSTRUCTORS

- *Constructor*

  `public` **ExpressionInt**`( )`

  Initializes a new instance of the ”ExpressionInt” class.

  - **Parameters**
    - `integer` - The integer value of this expression.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *RunCSPBuildExpr*

  `assembly` **Casper.Data.CSPExpr RunCSPBuildExpr**`( )`

  Abstract method which runs CSPBuildExpr in CLab.CSP.CSPExpressionBuilder Double dispatch pattern

  - **Parameters**
    * `builder` - The CSPExpressionBuilder.

- *ToString*

  `public` **string ToString**`( )`

Returns a textual representation of the expression.

- *Type2oper*

  `public` **string Type2oper** `( )`

  Converts the type to an operator sign for printout

  - **Parameters**
    - `type` -

CLASS **ExpressionNotNeg**

An negated expression consisting of a left expression and negation type Implements the abstract Expression class.

DECLARATION

```
public class ExpressionNotNeg
    : Expression
```

PROPERTIES

- *Left*

  `assembly` **CLab.Data.Expression Left** `{ get; }`

  Gets the left expression.

- *Type*

  `public` **CLab.Common.ExprType Type** `{ get; }`

  Returns the type of this expression

CONSTRUCTORS

- *Constructor*

  `public` **ExpressionNotNeg**`( )`

  Initializes a new instance of the ”ExpressionNotNeg” class.

  - **Parameters**
    - `type` - The operator.
    - `left` - The left expression.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *RunCSPBuildExpr*

  `assembly` **Casper.Data.CSPExpr RunCSPBuildExpr**`( )`

  Abstract method which runs CSPBuildExpr in CLab.CSP.CSPExpressionBuilder Double dispatch pattern

  - **Parameters**

    * `builder` - The CSPExpressionBuilder.

- *ToString*

  `public` **string ToString**`( )`

  Returns a textual representation of the not-neg-expression

- *Type2oper*

  `public` **string Type2oper**`( )`

  Converts the type to an operator sign for printout

  - **Parameters**
    * `type` -

CLASS **Symbols**

Class for checking if the problem is valid. Runs type and variable checks.

DECLARATION

```
public class Symbols
    : Object
```

PROPERTIES

- *AllVariables*

  `public` **Iesi.Collections.Set AllVariables** `{ get; }`

  Gets the set of all variables.

- *BoolVariables*

  `public` **Iesi.Collections.Set BoolVariables** `{ get;  set; }`

  Gets or sets the bool variables.

- *EnumerationVariables*

  `public` **Iesi.Collections.Set EnumerationVariables** `{ get; }`

  Gets the enumeration variables.

- *RangeVariables*

  `public` **Iesi.Collections.Set RangeVariables** `{ get;  set; }`

  Gets or sets the range variables.

CONSTRUCTORS

- *Constructor*

  `public` **Symbols**`( )`

  Initializes a new instance of the ”Symbols” class.

- **Parameters**
  - `cp` - The CP object.

METHODS

- *CheckExpressionType*

  `public` **void CheckExpressionType**`( )`

  Checks if the expressions have valid types.

  - **Parameters**
    - `expression` - The expression.

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *FillAndCheckType*

  `public` **void FillAndCheckType**`( )`

  Fills the type list after the type has been checked.

- *FillAndCheckVar*

  `public` **void FillAndCheckVar**`( )`

  Fills the variable list after the variable has been checked.

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MakeSetOfList*

  `public` **Iesi.Collections.Set MakeSetOfList**`( )`

  Makes a set of a list.

  - **Parameters**
    - `list` - The original list.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **Type**

The Type class represents a CLab type. There are two variants of Type objects: 1. Type(String typeName, ArrayList states) contains a enumeration of valid states for the type. 2. Type(String typeName, int startOfRange, int endOfRange) contains a set range.

DECLARATION

public class Type
: Object

PROPERTIES

- *TypeName*

  `public` **string TypeName** `{ get; set; }`

  Gets or sets the string representation of the name of the type.

CONSTRUCTORS

- *Constructor*

  `public` **Type**`( )`

  Initializes a new instance of the ”Type” class.

  - **Parameters**
    - `typeName` - Name of the type.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns the name of the type.

CLASS **TypeEnum**

The Type class represents a CLab type CSPTypeEnum represents an enumeration type with a list of valid states (domain) Implements the abstract Type class.

DECLARATION

```
public class TypeEnum
    : Type
```

PROPERTIES

- *EnumeratorsList*

  `public` **System.Collections.Generic.List{System.String} EnumeratorsList** `{ get; }`

  Gets the list of valid states for the type.

- *GetDomain*

  `public` **string GetDomain** `{ get; }`

  Gets a string representation of the valid states for the type in the form {value1, value2, value3, ... , valueN}

- *TypeName*

  `public` **string TypeName** `{ get;  set; }`

  Gets or sets the string representation of the name of the type.

CONSTRUCTORS

- *Constructor*

  `public` **TypeEnum**`( )`

  Initializes a new instance of the ”TypeEnum” class.

  – **Parameters**

  * `typeName` - Name of the type.
  * `enumeratorsList` - List of valid states for the type.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns the names of the type and its valid states.

CLASS **TypeRange**

The Type class represents a CLab type TypeRange represents a range type. Implements the abstract Type class.

DECLARATION

```
public class TypeRange
    : Type
```

PROPERTIES

- *EndOfRange*

  `public` **int EndOfRange** `{ get; set; }`

  Gets or sets the integer value of the end of the range.

- *StartOfRange*

  `public` **int StartOfRange** `{ get; set; }`

  Gets or sets the integer value of the start of the range.

- *TypeName*

  `public` **string TypeName** `{ get; set; }`

  Gets or sets the string representation of the name of the type.

CONSTRUCTORS

- *Constructor*

  `public` **TypeRange**`( )`

  Initializes a new instance of the ”TypeRange” class.

  – **Parameters**
    * `typeName` - Name of the type.
    * `startOfRange` - The start of the range.
    * `endOfRange` - The end of the range.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj`-

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns the name of the type and its range.

CLASS **Variable**

The Variable class represents a CLab variable In addition to the types set by Type objects, CLab has an internal boolean type named bool

DECLARATION

public class Variable
: Object

PROPERTIES

- *TypeName*

  `public` **string TypeName** `{ get; set; }`

  Gets or sets the name of the variable type.

- *VariableName*

  `public` **string VariableName** `{ get; set; }`

  Gets or sets the name of the variable.

CONSTRUCTORS

- *Constructor*

  `public` **Variable**`( )`

  Initializes a new instance of the ”Variable” class.

  – **Parameters**
    * `typeName` - Name of the type to use for this variable.
    * `variableName` - Name of the variable.

METHODS

- *Equals*

`public` **bool Equals**`( )`

Determines whether the specified is equal to the current .

– **Parameters**

  ∗ `obj` -

• *Finalize*

`protected` **void Finalize**`( )`

Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

• *GetHashCode*

`public` **int GetHashCode**`( )`

Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

• *GetType*

`public` **System.Type GetType**`( )`

Gets the of the current instance.

• *MemberwiseClone*

`protected` **object MemberwiseClone**`( )`

Creates a shallow copy of the current .

• *ToString*

`public` **string ToString**`( )`

Returns a string representation of the name of the variable type and the variable name.

6.5 Namespace CLab.Exceptions

Namespace Contents	*Page*

Interfaces

Classes

6.5.1 Classes

CLASS **ClabException**

ClabException is a generic exception class used for separating exceptions which we throw and system exceptions. Looping through the inner exceptions should reflect the entire stack of messages and what lead to the exception.

DECLARATION

```
public class ClabException
      : Exception
```

PROPERTIES

- *Data*

  `public` **System.Collections.IDictionary Data** `{ get; }`

  Gets a collection of key/value pairs that provide additional, user-defined information about the exception.

- *HelpLink*

  `public` **string HelpLink** `{ get;  set; }`

  Gets or sets a link to the help file associated with this exception.

- *HResult*

  `protected` **int HResult** `{ get;  set; }`

  Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

- *InnerException*

  `public` **System.Exception InnerException** `{ get; }`

  Gets the instance that caused the current exception.

- *Message*

  `public` **string Message** `{ get; }`

  Gets a message that describes the current exception.

- *Source*

  `public` **string Source** `{ get;  set; }`

  Gets or sets the name of the application or the object that causes the error.

- *StackTrace*

  `public` **string StackTrace** `{ get; }`

  Gets a string representation of the frames on the call stack at the time the current exception was thrown.

- *TargetSite*

  `public` **System.Reflection.MethodBase TargetSite** `{ get; }`

  Gets the method that throws the current exception.

CONSTRUCTORS

- *Constructor*

  `public` **ClabException** `( )`

  Initializes a new instance of the ”ClabException” class.

  - **Parameters**
    - `message` - The exception message which describes the reason for the exception.
    - `inner` - The inner exception.

- *Constructor*

  `public` **ClabException** `( )`

  Initializes a new instance of the ”ClabException” class.

- **Parameters**
  - `message` - The exception message which describes the reason for the exception.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetBaseException*

  `public` **System.Exception GetBaseException**`( )`

  When overridden in a derived class, returns the that is the root cause of one or more subsequent exceptions.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetObjectData*

  `public` **void GetObjectData**`( )`

  When overridden in a derived class, sets the with information about the exception.

  - **Parameters**
    * `info` -
    * `context` -

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the runtime type of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Creates and returns a string representation of the current exception.

CLASS **ClabInternalErrorException**

Custom exception class used for internal errors. Receiver should either rethrow the exception or halt the execution of ClabSharp

DECLARATION

```
notpublic class ClabInternalErrorException
    : Exception
```

PROPERTIES

- *Data*

  `public` **System.Collections.IDictionary Data** `{ get; }`

  Gets a collection of key/value pairs that provide additional, user-defined information about the exception.

- *HelpLink*

  `public` **string HelpLink** `{ get;  set; }`

  Gets or sets a link to the help file associated with this exception.

- *HResult*

  `protected` **int HResult** `{ get;  set; }`

  Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

- *InnerException*

  `public` **System.Exception InnerException** `{ get; }`

  Gets the instance that caused the current exception.

- *Message*

  `public` **string Message** `{ get; }`

Gets a message that describes the current exception.

- *Source*

  `public` **string Source** `{ get;  set; }`

  Gets or sets the name of the application or the object that causes the error.

- *StackTrace*

  `public` **string StackTrace** `{ get; }`

  Gets a string representation of the frames on the call stack at the time the current exception was thrown.

- *TargetSite*

  `public` **System.Reflection.MethodBase TargetSite** `{ get; }`

  Gets the method that throws the current exception.

CONSTRUCTORS

- *Constructor*

  `public` **ClabInternalErrorException**`( )`

  Initializes a new instance of the ”ClabInternalErrorException” class.

  – **Parameters**

    ∗ `message` - The exception message which describes the reason for the exception.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  – **Parameters**

    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetBaseException*

  `public` **System.Exception GetBaseException**`( )`

  When overridden in a derived class, returns the that is the root cause of one or more subsequent exceptions.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetObjectData*

  `public` **void GetObjectData**`( )`

  When overridden in a derived class, sets the with information about the exception.

  - **Parameters**
    * `info` -
    * `context` -

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the runtime type of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Creates and returns a string representation of the current exception.

CLASS **ClabInvalidExpressionException**

Custom exception class triggered by invalid parsing during the building of expressions.

DECLARATION

```
notpublic class ClabInvalidExpressionException
      : Exception
```

PROPERTIES

- *Data*

  `public` **System.Collections.IDictionary Data** `{ get; }`

  Gets a collection of key/value pairs that provide additional, user-defined information about the exception.

- *HelpLink*

  `public` **string HelpLink** `{ get;  set; }`

  Gets or sets a link to the help file associated with this exception.

- *HResult*

  `protected` **int HResult** `{ get;  set; }`

  Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

- *InnerException*

  `public` **System.Exception InnerException** `{ get; }`

  Gets the instance that caused the current exception.

- *Message*

  `public` **string Message** `{ get; }`

  Gets a message that describes the current exception.

- *Source*

  `public` **string Source** { `get; set;` }

  Gets or sets the name of the application or the object that causes the error.

- *StackTrace*

  `public` **string StackTrace** { `get;` }

  Gets a string representation of the frames on the call stack at the time the current exception was thrown.

- *TargetSite*

  `public` **System.Reflection.MethodBase TargetSite** { `get;` }

  Gets the method that throws the current exception.

CONSTRUCTORS

- *Constructor*

  `public` **ClabInvalidExpressionException**`( )`

  Initializes a new instance of the ”ClabInvalidExpressionException” class.

  - **Parameters**
    * `expressionType` - Type of the expression.
    * `textValue` - The text value.

- *Constructor*

  `public` **ClabInvalidExpressionException**`( )`

  Initializes a new instance of the ”ClabInvalidExpressionException” class.

  - **Parameters**
    * `expressionType` - Type of the expression.
    * `integerValue` - The integer value.

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetBaseException*

  `public` **System.Exception GetBaseException** `( )`

  When overridden in a derived class, returns the that is the root cause of one or more subsequent exceptions.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetObjectData*

  `public` **void GetObjectData** `( )`

  When overridden in a derived class, sets the with information about the exception.

  - **Parameters**
    - `info` -
    - `context` -

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the runtime type of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Creates and returns a string representation of the current exception.

CLASS **ClabSymbolException**

Custom exception class used for internal errors. Receiver should either rethrow the exception or halt the execution of ClabSharp

DECLARATION

```
notpublic class ClabSymbolException
    : Exception
```

PROPERTIES

- *Data*

  `public` **System.Collections.IDictionary Data** `{ get; }`

  Gets a collection of key/value pairs that provide additional, user-defined information about the exception.

- *HelpLink*

  `public` **string HelpLink** `{ get;  set; }`

  Gets or sets a link to the help file associated with this exception.

- *HResult*

  `protected` **int HResult** `{ get;  set; }`

  Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

- *InnerException*

  `public` **System.Exception InnerException** `{ get; }`

  Gets the instance that caused the current exception.

- *Message*

  `public` **string Message** `{ get; }`

Gets a message that describes the current exception.

- *Source*

  `public` **string Source** `{ get;  set; }`

  Gets or sets the name of the application or the object that causes the error.

- *StackTrace*

  `public` **string StackTrace** `{ get; }`

  Gets a string representation of the frames on the call stack at the time the current exception was thrown.

- *TargetSite*

  `public` **System.Reflection.MethodBase TargetSite** `{ get; }`

  Gets the method that throws the current exception.

CONSTRUCTORS

- *Constructor*

  `public` **ClabSymbolException**`( )`

  Initializes a new instance of the ”ClabSymbolException” class.

  - **Parameters**
    - `message` - The exception message which describes the reason for the exception.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**

    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetBaseException*

  `public` **System.Exception GetBaseException**`( )`

  When overridden in a derived class, returns the that is the root cause of one or more subsequent exceptions.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetObjectData*

  `public` **void GetObjectData**`( )`

  When overridden in a derived class, sets the with information about the exception.

  - **Parameters**
    * `info` -
    * `context` -

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the runtime type of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Creates and returns a string representation of the current exception.

CLASS **ClabXMLParserException**

Custom exception class used for errors during XML parsing. ClabXmlParser throws this exception as a result when any one of these exceptions are caught in the parser: XmlExceptionXmlSchemaValidationExceptionSystem.IO.FileNotFoundExceptionInvalidOperationExceptionClabInvalidExpressionException

DECLARATION

```
notpublic class ClabXMLParserException
      : Exception
```

PROPERTIES

- *Data*

  `public` **System.Collections.IDictionary Data** `{ get; }`

  Gets a collection of key/value pairs that provide additional, user-defined information about the exception.

- *HelpLink*

  `public` **string HelpLink** `{ get;  set; }`

  Gets or sets a link to the help file associated with this exception.

- *HResult*

  `protected` **int HResult** `{ get;  set; }`

  Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

- *InnerException*

  `public` **System.Exception InnerException** `{ get; }`

  Gets the instance that caused the current exception.

- *Message*

`public` **string Message** `{ get; }`

Gets a message that describes the current exception.

- *Source*

  `public` **string Source** `{ get;  set; }`

  Gets or sets the name of the application or the object that causes the error.

- *StackTrace*

  `public` **string StackTrace** `{ get; }`

  Gets a string representation of the frames on the call stack at the time the current exception was thrown.

- *TargetSite*

  `public` **System.Reflection.MethodBase TargetSite** `{ get; }`

  Gets the method that throws the current exception.

CONSTRUCTORS

- *Constructor*

  `public` **ClabXMLParserException**`( )`

  Initializes a new instance of the ”ClabXMLParserException” class.

  - **Parameters**
    * `message` - The exception message which describes the reason for the exception.
    * `inner` - The inner exception.

- *Constructor*

  `public` **ClabXMLParserException**`( )`

  Initializes a new instance of the ”ClabXMLParserException” class.

  - **Parameters**

    * `message` - The exception message which describes the reason for the exception.

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetBaseException*

  `public` **System.Exception GetBaseException** `( )`

  When overridden in a derived class, returns the that is the root cause of one or more subsequent exceptions.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetObjectData*

  `public` **void GetObjectData** `( )`

  When overridden in a derived class, sets the with information about the exception.

  - **Parameters**

  - `info` -
  - `context` -

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the runtime type of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Creates and returns a string representation of the current exception.

CLASS **TokenParserException**

Exception for error handling during token parsing

DECLARATION

```
public class TokenParserException
      : Exception
```

PROPERTIES

- *Data*

  `public` **System.Collections.IDictionary Data** `{ get; }`

  Gets a collection of key/value pairs that provide additional, user-defined information about the exception.

- *HelpLink*

  `public` **string HelpLink** `{ get;  set; }`

  Gets or sets a link to the help file associated with this exception.

- *HResult*

  `protected` **int HResult** `{ get;  set; }`

  Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

- *InnerException*

  `public` **System.Exception InnerException** `{ get; }`

  Gets the instance that caused the current exception.

- *Message*

  `public` **string Message** `{ get; }`

  Gets a message that describes the current exception.

- *Source*

  `public` **string Source** `{ get;  set; }`

  Gets or sets the name of the application or the object that causes the error.

- *StackTrace*

  `public` **string StackTrace** `{ get; }`

  Gets a string representation of the frames on the call stack at the time the current exception was thrown.

- *TargetSite*

  `public` **System.Reflection.MethodBase TargetSite** `{ get; }`

  Gets the method that throws the current exception.

CONSTRUCTORS

- *Constructor*

  `public` **TokenParserException**`( )`

  Initializes a new instance of the ”TokenParserException” class.

  – **Parameters**
    - ∗ `message` - The exception message which describes the reason for the exception.
    - ∗ `inner` - The inner exception.

- *Constructor*

  `public` **TokenParserException**`( )`

  Initializes a new instance of the ”TokenParserException” class.

  – **Parameters**
    - ∗ `message` - The exception message which describes the reason for the exception.

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetBaseException*

  `public` **System.Exception GetBaseException** `( )`

  When overridden in a derived class, returns the that is the root cause of one or more subsequent exceptions.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetObjectData*

  `public` **void GetObjectData** `( )`

  When overridden in a derived class, sets the with information about the exception.

  - **Parameters**
    - `info` -
    - `context` -

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the runtime type of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Creates and returns a string representation of the current exception.

6.6 Namespace CLab.Parsers

Namespace Contents *Page*

6.6.1 Classes

CLASS **CLabTextParser**

TextParser for converting from plain text CP to XML, or XML to CP Parsing from XML utilizes the ”ClabXmlParser” class.

DECLARATION

public class CLabTextParser
  : Object

CONSTRUCTORS

- *Constructor*

  `public` **CLabTextParser**`( )`

  Initializes a new instance of the class.

METHODS

- *CPtoXML*

  `public` **System.IO.MemoryStream CPtoXML**`( )`

  Parses plain text CP to XML format

  – **Parameters**
    * `input` - String with CP input to parse

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  – **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *XMLtoCP*

  `public` **string XMLtoCP**`( )`

  Parses XML input to plain text CP

  – **Parameters**
    * `filename` - Filename of XML input file

CLASS **ClabXmlParser**

Parses and validates the input XML file. Inserts types, variables and rules to the set CP object

DECLARATION

public class ClabXmlParser
: Object

CONSTRUCTORS

- *Constructor*

  `public` **ClabXmlParser**`( )`

  Initializes a new instance of the "ClabXmlParser" class.

  - **Parameters**
    * `cpObject` - The cp object which gets the data from the xml file.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *InitializeParser*

  `public` **void InitializeParser**`( )`

  Initializes the parser.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *Parse*

  `public` **void Parse**`( )`

  Parses the specified stream.

  - **Parameters**
    * `stream` - The stream.

- *Parse*

  `public` **void Parse**`( )`

  Parses the specified XML file.

  - **Parameters**
    * `xmlUrl` - URL for the XML file.

6.7 Namespace com.calitha.goldparser

6.7.1 Classes

CLASS **RuleException**

Internal exception class used by the parser engine

DECLARATION

public class RuleException
: Exception

PROPERTIES

- *Data*

  `public` **System.Collections.IDictionary Data** `{ get; }`

  Gets a collection of key/value pairs that provide additional, user-defined information about the exception.

- *HelpLink*

  `public` **string HelpLink** `{ get;  set; }`

  Gets or sets a link to the help file associated with this exception.

- *HResult*

  `protected` **int HResult** `{ get;  set; }`

  Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

- *InnerException*

  `public` **System.Exception InnerException** `{ get; }`

  Gets the instance that caused the current exception.

- *Message*

`public` **string Message** `{ get; }`

Gets a message that describes the current exception.

- *Source*

  `public` **string Source** `{ get;  set; }`

  Gets or sets the name of the application or the object that causes the error.

- *StackTrace*

  `public` **string StackTrace** `{ get; }`

  Gets a string representation of the frames on the call stack at the time the current exception was thrown.

- *TargetSite*

  `public` **System.Reflection.MethodBase TargetSite** `{ get; }`

  Gets the method that throws the current exception.

CONSTRUCTORS

- *Constructor*

  `public` **RuleException**`( )`

  Initializes a new instance of the ”RuleException” class.

  - **Parameters**
    - `message` - The exception message which describes the reason for the exception.

- *Constructor*

  `public` **RuleException**`( )`

  Initializes a new instance of the ”RuleException” class.

  - **Parameters**

    * `message` - The exception message which describes the reason for the exception.
    * `inner` - The inner exception.

- *Constructor*

  `protected` **RuleException**`( )`

  Initializes a new instance of the "RuleException" class.

  – **Parameters**

    * `info` - The "System.Runtime.Serialization.SerializationInfo" that holds the serialized object data about the exception being thrown.
    * `context` - The "System.Runtime.Serialization.StreamingContext" that contains contextual information about the source or destination.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  – **Parameters**

    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetBaseException*

  `public` **System.Exception GetBaseException**`( )`

  When overridden in a derived class, returns the that is the root cause of one or more subsequent exceptions.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetObjectData*

  `public` **void GetObjectData**`( )`

  When overridden in a derived class, sets the with information about the exception.

  - **Parameters**
    - `info` -
    - `context` -

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the runtime type of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Creates and returns a string representation of the current exception.

CLASS SymbolException

Internal exception class used by the parser engine

DECLARATION

```
public class SymbolException
    : Exception
```

PROPERTIES

- *Data*

  `public` **System.Collections.IDictionary Data** `{ get; }`

  Gets a collection of key/value pairs that provide additional, user-defined information about the exception.

- *HelpLink*

  `public` **string HelpLink** `{ get;  set; }`

  Gets or sets a link to the help file associated with this exception.

- *HResult*

  `protected` **int HResult** `{ get;  set; }`

  Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

- *InnerException*

  `public` **System.Exception InnerException** `{ get; }`

  Gets the instance that caused the current exception.

- *Message*

  `public` **string Message** `{ get; }`

  Gets a message that describes the current exception.

- *Source*

  `public` **string Source** `{ get;  set; }`

  Gets or sets the name of the application or the object that causes the error.

- *StackTrace*

  `public` **string StackTrace** `{ get; }`

  Gets a string representation of the frames on the call stack at the time the current exception was thrown.

- *TargetSite*

  `public` **System.Reflection.MethodBase TargetSite** `{ get; }`

  Gets the method that throws the current exception.

CONSTRUCTORS

- *Constructor*

  `public` **SymbolException**`( )`

  Initializes a new instance of the ”SymbolException” class.

  - **Parameters**
    - `message` - The exception message which describes the reason for the exception.

- *Constructor*

  `public` **SymbolException**`( )`

  Initializes a new instance of the ”SymbolException” class.

  - **Parameters**
    - `message` - The exception message which describes the reason for the exception.
    - `inner` - The inner exception.

- *Constructor*

  `protected` **SymbolException**`( )`

  Initializes a new instance of the ”SymbolException” class.

  - **Parameters**
    - `info` - The ”System.Runtime.Serialization.SerializationInfo” that holds the serialized object data about the exception being thrown.
    - `context` - The ”System.Runtime.Serialization.StreamingContext” that contains contextual information about the source or destination.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetBaseException*

  `public` **System.Exception GetBaseException**`( )`

  When overridden in a derived class, returns the that is the root cause of one or more subsequent exceptions.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetObjectData*

  `public` **void GetObjectData**`( )`

  When overridden in a derived class, sets the with information about the exception.

  - **Parameters**
    - `info` -
    - `context` -

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the runtime type of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Creates and returns a string representation of the current exception.

CLASS **TokenParser**

Token based parser using the GOLD Parsing system. Grammar tables for parsing are compiled from the parser grammar located in the doc folder. The compiled grammar is located in the Auxiliary folder and is integrated in the compiled binaries. This source file was initially generated by the GOLD Parsing system, based on grammar definition. Not all placeholder code has been replaced/removed to facilitate further use.

DECLARATION

```
public class TokenParser
    : Object
```

CONSTRUCTORS

- *Constructor*

  `public` **TokenParser**`( )`

  Initializes a new instance of the "TokenParser" class. Parsing from a file defined by .

  - **Parameters**
    - `filename` - Filename of input file for parsing.
    - `xw` - The XML writer to write output to.

- *Constructor*

  `public` **TokenParser**`( )`

  Initializes a new instance of the "TokenParser" class.

  - **Parameters**
    - `stream` - The stream to parse.
    - `xw` - The XML writer to write output to.

METHODS

- *CreateObjectFromNonterminal*

`public` **object CreateObjectFromNonterminal**`( )`

Manage a nonterminal token object

- **Parameters**
  - `token` - Token to manage

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *Parse*

  `public` **void Parse**`( )`

  Parses the specified source.

  - **Parameters**
    - `source` - The source in string representation.

- *ToString*

  `public` **string ToString**`( )`

  Returns a that represents the current .

Chapter 7

CaSPer API Reference

7.1 Namespace Casper

Namespace Contents *Page*

Interfaces

Classes

7.1.1 Classes

CLASS **Csp**

The interface of the Casper library. This class supports searching for valid domains, and reducing domains based on user selected domain values, using the CSP algorithm ”Select value forward checking”, and ”Generalized look ahead”.

DECLARATION

```
public class Csp
    : Object
```

PROPERTIES

- *CasperVarDoms*

  `public` **System.Collections.Generic.List{Casper.Data.CasperVarDom} CasperVarDoms** `{ get; }`

  Gets the variable and domain list.

- *CGraph*

  `public` **Casper.Data.ConstraintGraph CGraph** `{ get;  set; }`

  Gets or sets the constraint graph.

- *Expressions*

  `public` **Casper.Data.CSPExpressions Expressions** `{ get;  set; }`

  Gets or sets the expressions.

- *OmitTestOnSingleValuedDomains*

  `public` **bool OmitTestOnSingleValuedDomains** `{ get;  set; }`

  Gets or sets a value indicating whether single valued domains should be omited.

- *VariableOrdering*

  `public` **Casper.CspVariableOrdering VariableOrdering** `{ get; }`

  Gets the variable ordering.

CONSTRUCTORS

- *Constructor*

  `public` **Csp**`( )`

  Initializes a new instance of the ”Csp” class. The default Implementation of ”IConsistent” is used.

  - **Parameters**
    - `expressions` - The expressions.
    - `casperVarDoms` - The list with variables/domains.

- *Constructor*

  `public` **Csp**`( )`

  Initializes a new instance of the ”Csp” class. ”CasperVarDom” objects must be added afterwards, to use the ”valid domains” methods. The default implementation of ”IConsistent” is used.

  - **Parameters**
    - `expressions` - The expressions.

- *Constructor*

  `public` **Csp**`( )`

  Initializes a new instance of the ”Csp” class. ”CasperVarDom” objects must be added, and a consistent implementation of ”IConsitent” must be set afterwards, to use the ”valid domains” methods.

METHODS

- *AddCasperVarDom*

  `public` **int AddCasperVarDom**`( )`

  Adds a new variable and domain to the problem.

  - **Parameters**
    - `vardom` - The new variable and domain

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

`protected` **object MemberwiseClone** `( )`

Creates a shallow copy of the current .

- *SetVariableOrdering*

  `public` **void SetVariableOrdering** `( )`

  - **Parameters**
    - `varorder` -

- *ValidDomains*

  `public` **System.Collections.Generic.List{Casper.Data.CasperVarDom} ValidDomains** `( )`

  This method runs the forward checking algorithm in the ”GeneralizedLookahead” class. If the data field omitTestOnSIgnleDomains is set to true, one valued domain variables are jumped over.

- *ValidDomainsUserChoice*

  `public` **System.Collections.Generic.List{Casper.Data.CasperVarDom} ValidDomainsUserChoice** `( )`

  - **Parameters**
    - `var` -
    - `domainVal` -

CLASS **StaticData**

A class for Caspers static data.

DECLARATION

```
public class StaticData
    : Object
```

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

7.2 Namespace Casper.Algorithm

Namespace Contents *Page*

Interfaces

Classes

Consistent 194

CaSPers default implementation of the IConsistent interface

GeneralizedLookahead 198

Implementation of "Generalized lookahead" with "Select value forward checking".

LookAheadVarDom 202

The class which stores the information of a variable and its domaine values. Used by "GeneralizedLookahead".

LookaheadVariables 205

Class storing the variables and domaines used by "GenerelizedLookahed".

RemovedValues 207

Class for keeping track of which variable removed which values from the domain of this variable. This is used under resetting the variable's domain when backtracking, so that we are sure we only reset the domain to the values removed by the

7.2.1 Classes

CLASS **Consistent**

CaSPers default implementation of the IConsistent interface

DECLARATION

public class Consistent
: Object

CONSTRUCTORS

- *Constructor*

  `public` **Consistent**`( )`

  Initializes a new instance of the ”ConsistentImpl” class.

  – **Parameters**
    * `expressions` - The expressions to be used as rules.

METHODS

- *AssignedVarsContainsAll*

  `public` **bool AssignedVarsContainsAll**`( )`

  Verifies whether the assigned variables contains all the variables in the rule or not

  – **Parameters**
    * `vars` - Set of variables in the current rule

- *ConsistentCheckExpression*

  `public` **int ConsistentCheckExpression**`( )`

  Returns the currently selected value for the variable associated in the expression

  – **Parameters**

    * `rule` - Current expression

- *ConsistentCheckExpression*

  `public` **int ConsistentCheckExpression**`( )`

  Returns the currently selected value for the variable associated in the expression

  - **Parameters**
    * `rule` - Current expression

- *ConsistentCheckExpression*

  `public` **int ConsistentCheckExpression**`( )`

  Returns the currently selected value for the variable associated in the expression

  - **Parameters**
    * `rule` - Current expression

- *ConsistentCheckExpression*

  `public` **int ConsistentCheckExpression**`( )`

  Returns the Int/Const value of the terminal expression

  - **Parameters**
    * `rule` - Current expression

- *ConsistentCheckExpression*

  `public` **int ConsistentCheckExpression**`( )`

  Returns the result of the negation.

  - **Parameters**
    * `rule` - Current expression

- *ConsistentCheckExpression*

  `public` **int ConsistentCheckExpression**`( )`

  Returns the result of binary expressions

- **Parameters**
  - `rule` - Current expression

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *IsConsistent*

  `public` **bool IsConsistent**`( )`

  Main method for Consistency checks

  - **Parameters**
    - `allAssignedVars` - All assigned variables, including currAss and nextAss. Hashtable of LookAheadVarDom objects
    - `currAss` - Current assignment

  * `nextAss` - Next assignment

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **GeneralizedLookahead**

Implementation of ”Generalized lookahead” with ”Select value forward checking”.

DECLARATION

```
public class GeneralizedLookahead
      : Object
```

FIELDS

- *ASCENDINGORDER*

  `public` **int ASCENDINGORDER**

  Constant which represent ascending variable ordering.

- *MINWIDTHORDER*

  `public` **int MINWIDTHORDER**

  Constant which represent the minimum width variable ordering.

PROPERTIES

- *VariableOrdering*

  `public` **Casper.CspVariableOrdering VariableOrdering** `{ get;  set; }`

  Gets or sets the variable ordering.

CONSTRUCTORS

- *Constructor*

  `public` **GeneralizedLookahead**`( )`

  Initializes a new instance of the ”GeneralizedLookahead” class.

  – **Parameters**

    ∗ `expressions` - The expressions.
    ∗ `variableOrdering` - The variable ordering.
    ∗ `cGraph` - The constraint graph.

METHODS

- *BacktrackDomains*

  `public` **void BacktrackDomains**`( )`

  This method backtracks the changes the variable ”villain” executed on the other variables’ domains.

  – **Parameters**
    * `villain` - The villain.

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  – **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *ForwardCheck*

  `public` **bool ForwardCheck**`( )`

  Loops through future variables with respect to the current variable. This method checks if one of the future variables’ domains is empty with the current assignment.

  – **Parameters**
    * `currentVar` - The variable we are currently investigating.
    * `i` - The index ”currentVar” has in the variable list

- *GetHashCode*

  `public` **int GetHashCode**`( )`

Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *LookAhead*

  `public` **System.Collections.Generic.List{Casper.Algorithm.LookAheadVarDom} LookAhead** `( )`

  The Generalized Lookahead routine from Dechter page 133

  - **Parameters**
    - `varDoms` - List of CasperVarDom objects, with variable and domain data

- *MakeLookAheadVarDoms*

  `public` **Casper.Algorithm.LookaheadVariables MakeLookAheadVarDoms** `( )`

  Method for making LookAheadVar objects from CasperVarDom objects, for internal use in this class. Different variable orderings can be made with the constraint graph.

  - **Parameters**
    - `casperVarDoms` - List of ”CasperVarDom” objects.
    - `varOrder` - The var order.
    - `Graph` - The constraint graph.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *PruneNextDomain*

  `public` **int PruneNextDomain** `( )`

  Checks each domain value for the future variables domains in forwardCheck Here we do the actual consistency check with the provided consistency implementation.

- **Parameters**
  - `currentVar` - Currently checked variable
  - `nextVar` - The variable and domain the method runs consistency check on

• *Remove*

`public` **void Remove**`( )`

Updates the AllRemovedValues-structure with the removed value of variable ”victim” caused by variable ”villain”.

- **Parameters**
  - `victim` - The victim.
  - `villain` - The villain.

• *SelectValue*

`public` **System.Nullable{System.Int32} SelectValue**`( )`

Implementation of Select Value Forward Checking from Dechter page 134 Internal method used by the LookAhead method

- **Parameters**
  - `var` - Variable to be checked
  - `i` - The index ”var” has in the variable list

CLASS **LookAheadVarDom**

The class which stores the information of a variable and its domaine values. Used by ”GeneralizedLookahead”.

DECLARATION

```
public class LookAheadVarDom
    : Object
```

PROPERTIES

- *DomainValues*

  `public` **Iesi.Collections.Set DomainValues** `{ get;  set; }`

  Gets or sets the domain values.

- *SelectedValue*

  `public` **System.Nullable{System.Int32} SelectedValue** `{ get; }`

  Gets the current selected value of the variable

- *VarID*

  `public` **int VarID** `{ get;  set; }`

  Gets og sets the variable ID.

CONSTRUCTORS

- *Constructor*

  `public` **LookAheadVarDom**`( )`

  Initializes a new instance of the ”LookAheadVarDom” class.

  – **Parameters**
    ∗ `varID` - The variable ID.
    ∗ `domainValues` - The domain values.

METHODS

- *AddToRemoved*

  `public` **void AddToRemoved**`( )`

  – **Parameters**

    ∗ `val` -

- *DomainSize*

  `public` **int DomainSize**`( )`

  Gets the domain size.

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  – **Parameters**

    ∗ `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetNextValue*

  `public` **System.Nullable{System.Int32} GetNextValue**`( )`

  - **Parameters**
    * `startFromBeginning` -

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *ResetDomain*

  `public` **void ResetDomain** `( )`

  Method used by the algorithm to reset the domain during backtracking.

  - **Parameters**
    * `removedVals` - The removed values which should be reset.

- *UpdateDomainWithRemoved*

  `public` **void UpdateDomainWithRemoved** `( )`

  Updates the domain with removed values in the removed set. The removed set is emptied.

CLASS **LookaheadVariables**

Class storing the variables and domaines used by ”GenerelizedLookahed”.

DECLARATION

```
public class LookaheadVariables
      : Object
```

PROPERTIES

- *VarDoms*

  `public` **System.Collections.Hashtable VarDoms** `{ get; set; }`

  Gets or sets the variables and domaines.

- *VarOrder*

  `public` **Casper.Algorithm.VariableOrdering.IVariableOrdering VarOrder** `{ get; set; }`

  Gets or sets the variable ordering.

CONSTRUCTORS

- *Constructor*

  `public` **LookaheadVariables**`( )`

  Initializes a new instance of the ”LookaheadVariables” class.

  – **Parameters**

    ∗ `varOrderType` - The variable ordering
    ∗ `cGraph` - The constraint graph.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *GetVar*

  `public` **Casper.Algorithm.LookAheadVarDom GetVar**`( )`

  Gets the variable which belongs to number in queue.

  - **Parameters**
    - `numberInQueue` - The number in queue.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **RemovedValues**

Class for keeping track of which variable removed which values from the domain of this variable. This is used under resetting the variable's domain when backtracking, so that we are sure we only reset the domain to the values removed by the

DECLARATION

```
public class RemovedValues
      : Object
```

PROPERTIES

- *RemValStructure*

  `public` **System.Collections.Hashtable RemValStructure** `{ get;  set; }`

  Gets or sets the removed values structure.

CONSTRUCTORS

- *Constructor*

  `public` **RemovedValues**`( )`

  Initializes a new instance of the "RemovedValues" class.

METHODS

- *Clear*

  `public` **void Clear**`( )`

  Clears the removed value structure.

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *Insert*

  `public` **void Insert** `( )`

  Inserts the specified removed value. The key is the specified variable.

  - **Parameters**
    * `removedValue` - The removed value.
    * `var` - The variable.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *Remove*

  `public` **void Remove** `( )`

Sets the value at key ”var” in the remValStructure to be null.

– **Parameters**

  ∗ `var` - The variable

7.3 Namespace Casper.Algorithm.VariableOrdering

Namespace Contents *Page*

7.3.1 Interfaces

INTERFACE **IVariableOrdering**

Interface for variable ordering. The algorithm uses the GetNextVar method.

DECLARATION

public interface IVariableOrdering

:

METHODS

- *GetNextVar*

  `public` **int GetNextVar**`( )`

  All ordering implementations has to have a method for getting the next variable in relation to the passed in number in queue. This method is used by the algorithm.

  - **Parameters**
    - `numberInQueue` - The number in queue.

7.3.2 Classes

CLASS **MinimumWidthOrder**

Class for supporting minimum width ordering.

DECLARATION

public class MinimumWidthOrder

: Object

CONSTRUCTORS

- *Constructor*

`public` **MinimumWidthOrder**`( )`

Initializes a new instance of the ”MinimumWidthOrdering” class.

- **Parameters**
  * `cGraph` - The constraint graph.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetNextVar*

  `public` **int GetNextVar**`( )`

  Gets the next variable according to the minimum width ordering.

  - **Parameters**
    * `numberInQueue` - The number in queue.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

CLASS **StaticVarOrder**

An implementation of a static variable order, that is, the variable ordering equals the ordering given in the cp file.

DECLARATION

public class StaticVarOrder
: Object

CONSTRUCTORS

- *Constructor*

  `public` **StaticVarOrder**`( )`

  Initializes a new instance of the class.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  – **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetNextVar*

  `public` **int GetNextVar**`( )`

  Since the variables are stored increasingly, we return the number from input without any calculation.

  - **Parameters**
    - `numberInQueue` - The number in queue.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

7.4 Namespace Casper.Data

7.4.1 Classes

CLASS **AdjacencyList**

A class representing a list of adjacent nodes in a Constraint Graph.

DECLARATION

public class AdjacencyList
  : Object

PROPERTIES

- *Variables*

  `public` **Iesi.Collections.Set Variables** `{ get;  set; }`

  Gets or sets the list of adjacent nodes.

- *VariableWidth*

  `public` **int VariableWidth** `{ get;  set; }`

  Gets or sets the width of the variable.

CONSTRUCTORS

- *Constructor*

  `public` **AdjacencyList**`( )`

  Initializes a new instance of the ”AdjacencyList” class.

METHODS

- *AddAllVars*

  `public` **void AddAllVars**`( )`

  Adds a set of variable ids to the adjacency list.

  - **Parameters**
    * `vars` - The set of variables to add to the adjacency list.

- *AddVar*

  `public` **void AddVar** `( )`

  Adds a single variable id to the adjacency list.

  - **Parameters**
    * `varID` - The variable id to add to the adjacency list.

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *UpdateVarWidth*

  `public` **void UpdateVarWidth** `( )`

  Updates the width of the variable.

CLASS **CasperVarDom**

A class with variable and domain information, used by the ”Csp” class.

DECLARATION

public class CasperVarDom
: Object

PROPERTIES

- *DomainValues*

  `public` **System.Collections.Generic.List{System.Int32} DomainValues** `{ get; set; }`

  Gets or sets the domain values.

- *ValidValues*

  `public` **Iesi.Collections.Set ValidValues** `{ get;  set; }`

  Gets or sets the valid values.

- *VarID*

  `public` **int VarID** `{ get;  set; }`

  Gets or sets the variable ID.

CONSTRUCTORS

- *Constructor*

  `public` **CasperVarDom**`( )`

  Initializes a new instance of the ”CasperVarDom” class.

  – **Parameters**

    ∗ `varID` - The variable ID.
    ∗ `domainValues` - The domain values.

- *Constructor*

  `public` **CasperVarDom**`( )`

  Initializes a new instance of the ”CasperVarDom” class.

  - **Parameters**
    - `varID` - The variable ID.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString** `( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CasperVarDom”.

CLASS **ConstraintGraph**

A class which represents the constraint graph of a csp problem.

DECLARATION

public class ConstraintGraph

: Object

PROPERTIES

- *CGraph*

  `public` **System.Collections.Hashtable CGraph** `{ get; }`

  Gets the constraint graph.

CONSTRUCTORS

- *Constructor*

  `public` **ConstraintGraph**`( )`

  Initializes a new instance of the ”ConstraintGraph” class.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *UpdateAdjacencyList*

  `public` **void UpdateAdjacencyList**`( )`

  Updates the adjacency list representation of the graph by adding all neighbours of a variable to its adjacency list. Creates the adjacency list if it doesn't exist.

  - **Parameters**
    - `interconNodes` - A list of interconnected nodes

CLASS **CSPExpr**

An abstract class for CSP Expressions. All Expression classes implements this class.

DECLARATION

```
public class CSPExpr
    : Object
```

CONSTRUCTORS

- *Constructor*

  `public` **CSPExpr**`( )`

  Initializes a new instance of the ”CSPExpr” class.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  – **Parameters**
    * `obj-`

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **CSPExprBin**

DECLARATION

```
public class CSPExprBin
    : CSPExpr
```

PROPERTIES

- *Left*

  `public` **Casper.Data.CSPExpr Left** `{ get; }`

  Gets the left expression.

- *Oper*

  `public` **Casper.StaticData.Operators Oper** `{ get; }`

  Gets the operator.

- *Right*

  `public` **Casper.Data.CSPExpr Right** `{ get; }`

  Gets the right expression.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExprBin** `( )`

  Initializes a new instance of the ”CSPExprBin” class.

  – **Parameters**

    ∗ `left` - The left expression.
    ∗ `right` - The right expression.
    ∗ `oper` - The operator.

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString** `( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CSPExprBin”.

CLASS **CSPExpressions**

A class which keeps track of the expressions for a CSP problem.

DECLARATION

public class CSPExpressions

: Object

PROPERTIES

- *WrappedExprList*

  `public` **System.Collections.Generic.List{Casper.Data.CSPExprWrapper} WrappedExprList** `{ get; set; }`

  Gets or sets the wrappet expression list.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExpressions** `( )`

  Initializes a new instance of the "CSPExpressions" class.

METHODS

- *AddWrappedExpr*

  `public` **int AddWrappedExpr** `( )`

  Adds the wrappet expression.

  - **Parameters**
    - `wrappedExpr` - The wrappet expression.

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

- **Parameters**
  - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

CLASS **CSPExprNeg**

DECLARATION

```
public class CSPExprNeg
    : CSPExpr
```

PROPERTIES

- *Left*

  `public` **Casper.Data.CSPExpr Left** `{ get; }`

  Gets the expression.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExpNeg** `( )`

  Initializes a new instance of the ”CSPExprNeg” class.

  – **Parameters**
    * `left` - The left expression

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  – **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType** `( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone** `( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString** `( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CSPExprNeg”.

CLASS **CSPExprNot**

DECLARATION

```
public class CSPExprNotNeg
    : CSPExpr
```

PROPERTIES

- *Left*

  `public` **Casper.Data.CSPExpr Left** `{ get; }`

  Gets the expression.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExprNot**`( )`

  Initializes a new instance of the ”CSPExprNot” class.

  - **Parameters**
    * `left` - The left expression

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CSPExprNot”.

CLASS **CSPExprValue**

An abstract class for representing the value kind of expressions. Extends the abstract CSPExpr class.

DECLARATION

```
public class CSPExprValue
    : CSPExpr
```

PROPERTIES

- *Value*

  `public` **int Value** `{ get; set; }`

  Gets or sets the value.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExprValue**`( )`

  Initializes a new instance of the ”CSPExprValue” class.

  - **Parameters**
    * `value` - The value.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CSPExprValue”.

CLASS **CSPExprValueConst**

Implementation class for the constant type of value expressions. Implements the abstract CSP-ExprValue class.

DECLARATION

```
public class CSPExprValueConst
    : CSPExprValue
```

PROPERTIES

- *Value*

  `public` **int Value** `{ get;  set; }`

  Gets or sets the value.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExprValueConst**`( )`

  Initializes a new instance of the ”CSPExprValueConst” class.

  - **Parameters**
    - `constantID` - The constant ID.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CSPExprValueConst”.

CLASS **CSPExprValueInt**

Implementation class for the integer type of value expressions. Implements the abstract CSPExprValue class.

DECLARATION

```
public class CSPExprValueInt
      : CSPExprValue
```

PROPERTIES

- *Value*

  `public` **int Value** `{ get;  set; }`

  Gets or sets the value.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExprValueInt**`( )`

  Initializes a new instance of the ”CSPExprValueInt” class.

  - **Parameters**
    - `value` - The value.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CSPExprValueInt”.

CLASS **CSPExprVar**

An abstract class for representing the variable kind of expressions. Extends the abstract CSP-Expr class.

DECLARATION

```
public class CSPExprVar
      : CSPExpr
```

PROPERTIES

- *VarID*

  `public` **int VarID** `{ get;  set; }`

  Gets or sets the variable ID.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExprVar** `( )`

  Initializes a new instance of the ”CSPExprVar” class.

  - **Parameters**
    - `varID` - The var ID.

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    - `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CSPExpr”.

CLASS **CSPExprVarBool**

Implementation class for the boolean variable type of variable expressions. Implements the abstract CSPExprVar class.

DECLARATION

```
public class CSPExprVarBool
      : CSPExprVar
```

PROPERTIES

- *VarID*

  `public` **int VarID** `{ get;  set; }`

  Gets or sets the variable ID.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExprVarBool**`( )`

  Initializes a new instance of the ”CSPExprVarBool” class.

  – **Parameters**
    * `VarID` - The var ID.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  – **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CSPExprVarBool”.

CLASS **CSPExprVarEnum**

Implementation class for the enumeration variable type of variable expressions. Implements the abstract CSPExprVar class.

DECLARATION

public class CSPExprVarEnum
  : CSPExprVar

PROPERTIES

- *VarID*

  `public` **int VarID** `{ get;  set; }`

  Gets or sets the variable ID.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExprVarEnum** `( )`

  Initializes a new instance of the ”CSPExprVarEnum” class.

  - **Parameters**
    * `varID` - The var ID.

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CSPExprVarEnum”.

CLASS **CSPExprVarInt**

Implementation class for the integer variable type of variable expressions. Implements the CSPExprVar abstract class.

DECLARATION

```
public class CSPExprVarInt
    : CSPExprVar
```

PROPERTIES

- *VarID*

  `public` **int VarID** `{ get;  set; }`

  Gets or sets the variable ID.

CONSTRUCTORS

- *Constructor*

  `public` **CSPExprVarInt**`( )`

  Initializes a new instance of the ”CSPExprVarInt” class.

  - **Parameters**
    * `varID` - The var ID.

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Returns a ”System.String” that represents the current ”Casper.Data.CSPExprVarInt”.

CLASS **CSPExprWrapper**

Stores wrapped expression objects along with a set of its variables IDs. Used during consistency to filter out expressions which have variables which isn't set yet.

DECLARATION

```
public class CSPExprWrapper
    : Object
```

PROPERTIES

- *RuleExpression*

  `public` **Casper.Data.CSPExpr RuleExpression** `{ get; set; }`

  The contained expression object

- *VariablesInExpr*

  `public` **Iesi.Collections.Set VariablesInExpr** `{ get; set; }`

  Set of variables which the contained expression object consists of

CONSTRUCTORS

- *Constructor*

  `public` **CSPExprWrapper**`( )`

  Constructor.

  - **Parameters**
    * `anExpr` - An expression object representing a rule
    * `variablesInExpr` - Set of variable IDs used in the rule

METHODS

- *Equals*

  `public` **bool Equals**`( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize**`( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetHashCode*

  `public` **int GetHashCode**`( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

7.5 Namespace Casper.Exceptions

Namespace Contents	*Page*
Interfaces	

Classes

CasperException 251

Custom exception class used for exceptions thrown out of the library.

7.5.1 Classes

CLASS **CasperException**

Custom exception class used for exceptions thrown out of the library.

DECLARATION

```
public class CasperException
    : Exception
```

PROPERTIES

- *Data*

  `public` **System.Collections.IDictionary Data** `{ get; }`

  Gets a collection of key/value pairs that provide additional, user-defined information about the exception.

- *HelpLink*

  `public` **string HelpLink** `{ get;  set; }`

  Gets or sets a link to the help file associated with this exception.

- *HResult*

  `protected` **int HResult** `{ get;  set; }`

  Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

- *InnerException*

  `public` **System.Exception InnerException** `{ get; }`

  Gets the instance that caused the current exception.

- *Message*

`public` **string Message** `{ get; }`

Gets a message that describes the current exception.

- *Source*

  `public` **string Source** `{ get;  set; }`

  Gets or sets the name of the application or the object that causes the error.

- *StackTrace*

  `public` **string StackTrace** `{ get; }`

  Gets a string representation of the frames on the call stack at the time the current exception was thrown.

- *TargetSite*

  `public` **System.Reflection.MethodBase TargetSite** `{ get; }`

  Gets the method that throws the current exception.

CONSTRUCTORS

- *Constructor*

  `public` **CasperException** `( )`

  Initializes a new instance of the "CasperException" class.

  - **Parameters**
    - `message` - The exception message which describes the reason for the exception.

- *Constructor*

  `public` **CasperException** `( )`

  Initializes a new instance of the "CasperException" class.

  - **Parameters**

    * `message` - The exception message which describes the reason for the exception.
    * `inner` - The inner exception.

METHODS

- *Equals*

  `public` **bool Equals** `( )`

  Determines whether the specified is equal to the current .

  - **Parameters**
    * `obj` -

- *Finalize*

  `protected` **void Finalize** `( )`

  Allows an to attempt to free resources and perform other cleanup operations before the is reclaimed by garbage collection.

- *GetBaseException*

  `public` **System.Exception GetBaseException** `( )`

  When overridden in a derived class, returns the that is the root cause of one or more subsequent exceptions.

- *GetHashCode*

  `public` **int GetHashCode** `( )`

  Serves as a hash function for a particular type. is suitable for use in hashing algorithms and data structures like a hash table.

- *GetObjectData*

  `public` **void GetObjectData** `( )`

  When overridden in a derived class, sets the with information about the exception.

  - **Parameters**
    * `info` -
    * `context` -

- *GetType*

  `public` **System.Type GetType**`( )`

  Gets the runtime type of the current instance.

- *MemberwiseClone*

  `protected` **object MemberwiseClone**`( )`

  Creates a shallow copy of the current .

- *ToString*

  `public` **string ToString**`( )`

  Creates and returns a string representation of the current exception.

Chapter 8

GUI API Reference

8.1 Namespace ClabGui

Namespace Contents *Page*

8.1.1 Classes

CLASS **ClabSharpGui**

Main class of the GUI. Does nothing but to initialize the GUI.

DECLARATION

```
public class ClabSharpGui
        : Object
```

CLASS **ParameterBitmapSingleton**

A singleton for the different bitmaps representing the domain status for each domain value.

DECLARATION

```
public class ParameterBitmapSingleton
      : Object
```

CONSTRUCTORS

- *Constructor*

  `protected` **ParameterBitmapSingleton** `( )`

  Initializes a new instance of the ”ParameterBitmapSingleton” class.

METHODS

- *GetBitmap*

  `public` **System.Drawing.Bitmap GetBitmap** `( )`

  Gets the bitmap representing the passed in int value.

  – **Parameters**
    ∗ `STATUS` - The STATUS.

- *Instance*

  `public` **ClabGui.ParameterBitmapSingleton Instance** `( )`

  Creates a new instance if one doesn’t exist. Return the existing instance otherwize.

CLASS **Program**

This class is working between CLab and the GUI class, ”MainGui”. The searching part is run in its own thread, so the application won’t hang during a heavy search.

DECLARATION

```
public class Program
    : Object
```

PROPERTIES

- *Modus*

  `public` **CLab.CLabModus Modus** `{ get; set; }`

  Gets or sets the modus to use, BDD if modus = 0 or CSP if modus = 1.

- *ProblemFilename*

  `public` **string ProblemFilename** `{ get; set; }`

  Gets or sets the problem filename.

CONSTRUCTORS

- *Constructor*

  `public` **Program**`( )`

  Initializes a new instance of the ”Program” class.

- *Constructor*

  `public` **Program**`( )`

  Initializes a new instance of the ”Program” class.

  – **Parameters**

    ∗ `fileName` - Name of the file to be opened.

METHODS

- *AddDataToGui*

  `public` **void AddDataToGui**`( )`

  Adds the variable and domaine values data to GUI.

  - **Parameters**
    - `vd` - The vd.

- *ClabThreadIsAlive*

  `public` **bool ClabThreadIsAlive**`( )`

  Method for checking if CLab is running a search.

- *OpenFile*

  `public` **string OpenFile**`( )`

  Returns the content of a text file

  - **Parameters**
    - `fileName` - The file to get content from

- *PauseClabThreadIfRunning*

  `public` **bool PauseClabThreadIfRunning**`( )`

  Pauses the CLab thread if running, and resumes if paused.

- *ReadTextFromFile*

  `public` **string ReadTextFromFile**`( )`

  Reads the text from a file.

  - **Parameters**
    - `fileName` - Name of the file.

- *SaveFile*

  `public` **void SaveFile** `( )`

  Saves the file.

  - **Parameters**
    - `text` - The text.

- *SaveFileAs*

  `public` **void SaveFileAs** `( )`

  Saves the file as.

  - **Parameters**
    - `fileName` - Name of the file.
    - `text` - The text.

- *SetBDDCompileMethod*

  `public` **void SetBDDCompileMethod** `( )`

  Sets the BDD compile method.

  - **Parameters**
    - `compileMethod` - The compile method.

- *SetCSPVariableOrdering*

  `public` **void SetCSPVariableOrdering** `( )`

  Sets the CSP variable ordering.

  - **Parameters**
    - `varOrder` - The variable ordering.

- *StartClab*

  `public` **bool StartClab** `( )`

  Starts a new CLab instance, and fills the GUI with initial data.

- *StartClabInitialSearchThread*

  `public` **void** **StartClabInitialSearchThread**`( )`

  Starts the clab initial search thread.

- *StopClab*

  `public` **void** **StopClab**`( )`

  Sets the CLab reference to null.

- *StopClabThreadIfRunning*

  `public` **void** **StopClabThreadIfRunning**`( )`

  Stops the clab thread if running.

- *StripPathFromFilename*

  `public` **string** **StripPathFromFilename**`( )`

  Strips the path from filename.

  - **Parameters**
    - `filename` - A ”System.String” that represents the filename with full path

- *UpdateGuiDomains*

  `public` **void** **UpdateGuiDomains**`( )`

  Updates the GUI domains with the valid domains.

  - **Parameters**
    - `vd` - The valid domaines, ”CLab.ValidDomains”.

- *ValueChosenSearchThread*

  `public` **void** **ValueChosenSearchThread**`( )`

  Starts the clab value chosen search thread.

  - **Parameters**

    * `guiId` - The GUI variable ID, for chosen variable.
    * `domainVal` - The chosen domain value.

- *WriteCPToXml*

  `public` **void** **WriteCPToXml**`( )`

  Writes a xml file. Writes the CP text to XML.

  - **Parameters**
    * `fileName` - Name of the file.
    * `text` - The CP text.

- *WriteTextFile*

  `public` **void** **WriteTextFile**`( )`

  Writes a text file.

  - **Parameters**
    * `fileName` - Name of the file.
    * `text` - The text.

Bibliography