CONSTRAINED GOAL MODELING AND REASONING TOOL

USER MANUAL

(revision 1.0.1)

For CGM-Tool Version 1.0.0

`http://www.cgm-tool.eu/`

UNIVERSITY OF TRENTO - Italy
Information Engineering
and Computer Science Department

Contents

1 Getting Started

CGM-Tool is a modeling and reasoning tool support for constrained goal models[1]. It enables diagrammatic modeling of constrained goals, with automated well-formedness analysis and multi-objective lexicographical reasoning. CGM-Tool utilizes *OptiMathSAT* SMT solver[2] as a back-end engine for its reasoning functionality.

To start using CGM-Tool, first extract the contents of the downloaded archive file into any directory of your computer. Since the reasoning functionality of the tool principally relies on *OptiMathSAT* solver, installation of *OptiMathSAT* is required. For this, you need to download *OptiMathSAT* solver from `http://optimathsat.disi.unitn.it/` and extract its content into **OptiMathSAT folder** of the extracted tool. After that, run the launcher (*CGM-Tool*) [1], consequently a splash screen with the logo and version of the tool will appear in the middle of your screen indicating that the program is loading (Figure 1).

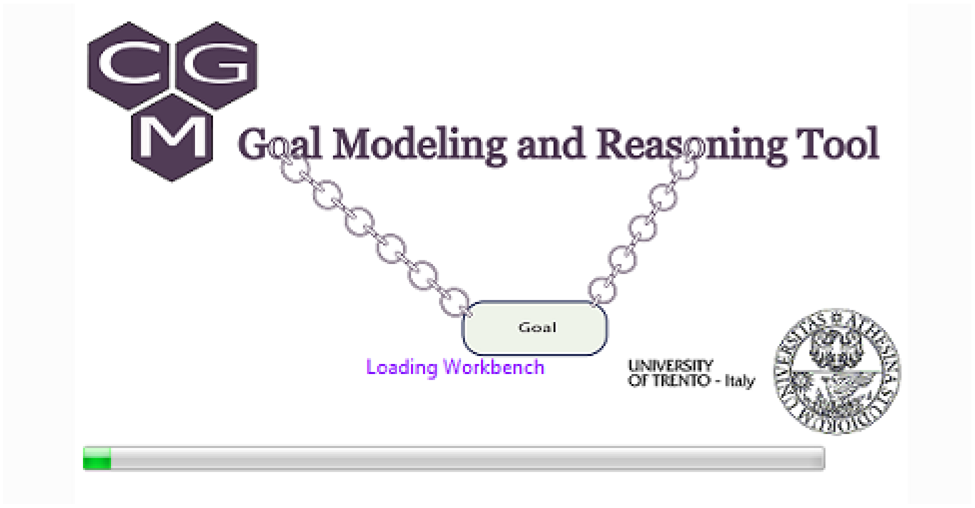
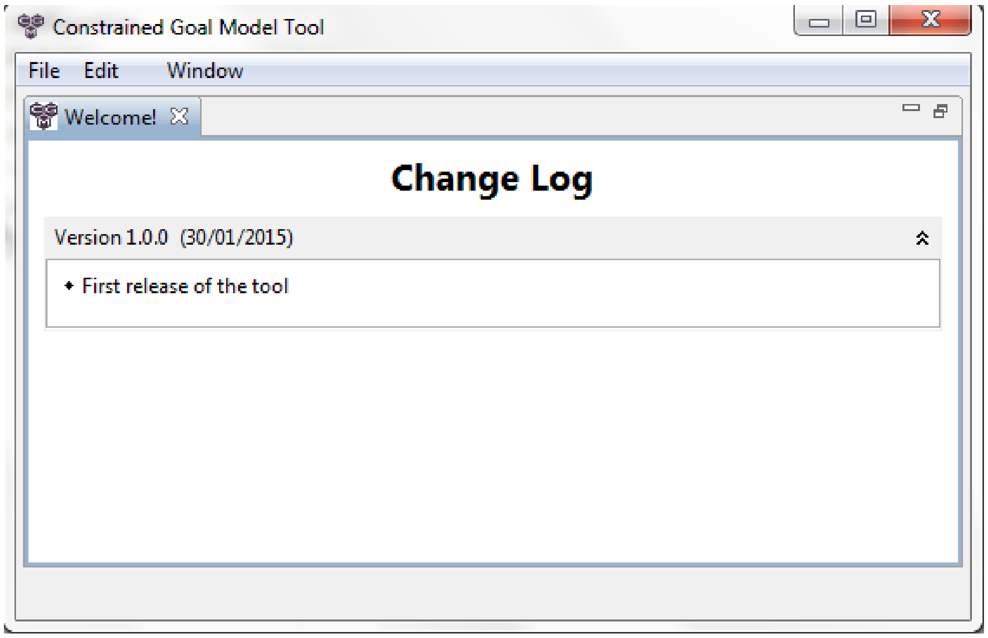

CGM
Goal Modeling and Reasoning Tool
Goal
Loading Workbench
UNIVERSITY OF TRENTO - Italy

Figure 1: CGM-Tool splash screen

Subsequently, you will be asked to read and accept the terms of use. To start using the tool select the **license agreement checkbox** on the bottom of the page and press **I Agree** button, you can also click on **Decline** button in order to turn down the license agreement and exit from the program. After you agree with the license, the tool will open a change log window (Figure 2 and present you lists of recent changes made on the tool. The log window can be closed by the little **X** button right next to the **Change Log** tab.

[1]CGM-Tool is Eclipse RCP based application, because of this no prior set-up/installation is required



Change Log

Version 1.0.0 (30/01/2015)

- First release of the tool

Figure 2: CGM-Tool change log

Minimum requirement!

- To run CGM-Tool you need to install **Java 7** or more.

Important for Linux based OS Users!

CGM-Tool and OptiMathSAT solver require **execute privilege** to function correctly. Please follow the following steps to fulfill this need :

[I] For CGM-Tool:

(a) Right click on the **launcher** (i.e. CGM-Tool) and select **"Property"**.

(b) Then, on **permission** tab, check **"Allow executing file as a program"** check box.

[II] For OptiMathSAT

(a) First go to **bin** folder of the extracted **OptiMathSAT** folder (which resides inside "`.../CGMTool_Linux_64bit/OptiMathSAT/optimathsat-1.3.1-linux-l64/bin''`.

(b) Inside **bin** directory you will find **optimathsat** file, right click on it and choose **property**. Then, on **permission** tab, check **"Allow executing file as a program"** check box.

2 GUI and Working Environment

This section describes the basic GUI parts of CGM-Tool, and also provides a general overview of the tool working environment. Figure 3 outlines the general appearance of the tool.

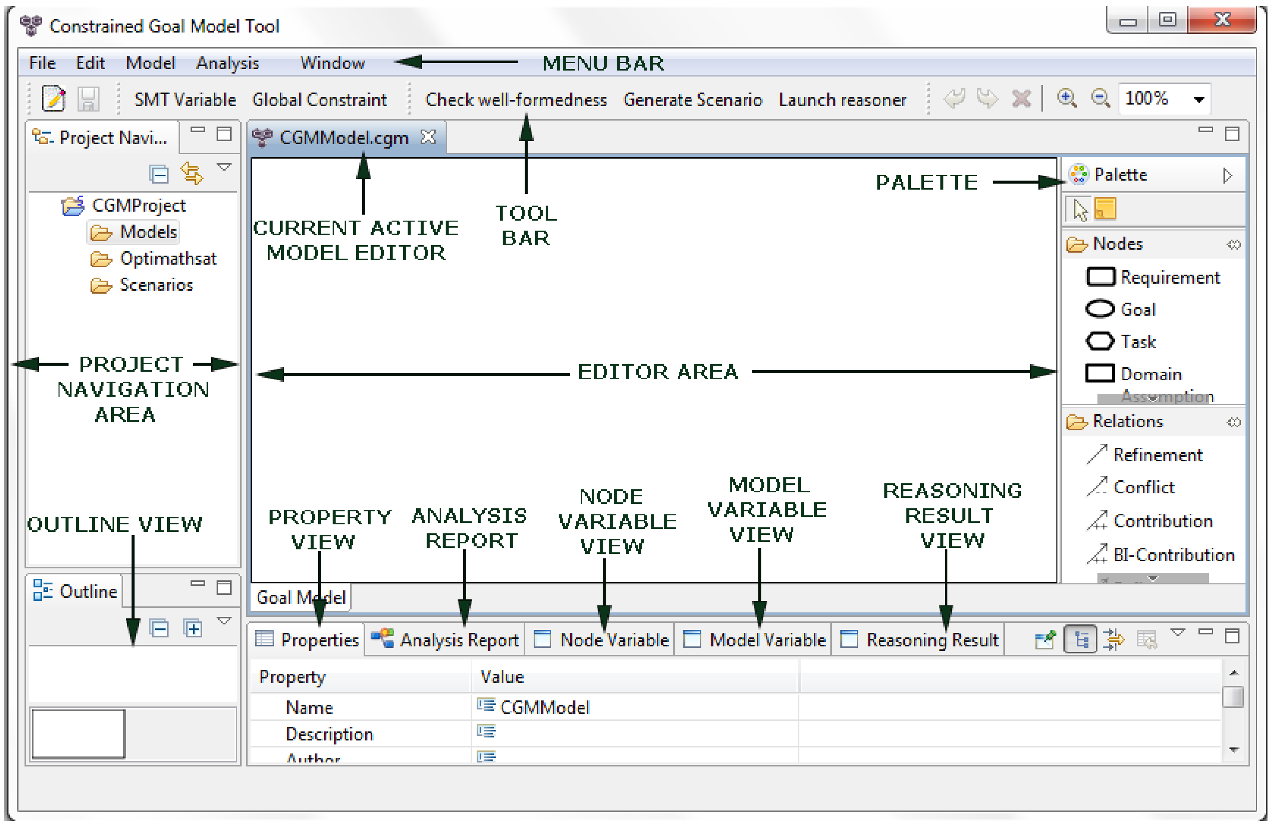


Figure 3: CGM-Tool Overview

2.1 CGM-Tool GUI

The graphical user interface of CGM-Tool is composed of menus, editor area, palette and several different views. The following sub-sections will explain parts of the GUI one by one.

2.1.1 Menus

Commands, which are scattered over the tool menu bar and tool bar are the main interfaces to access the primary functions of CGM-Tool. The succeeding sub-sections will introduce these commands and their use.

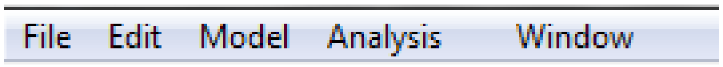


Figure 4: CGM-Tool menu bar

[I] Menu Bar

CGM-Tool menu bar (Figure 4) is a collection of the following menus:

(a) File Menu:

File menu (Figure 5) of the tool contains commands such as **Create New CGM Project**, **Create New CGM Model** and **Save** options, which are mainly used for handling of CGM models and projects. Furthermore, it offers commands for closing the active editor and for exiting from the application.

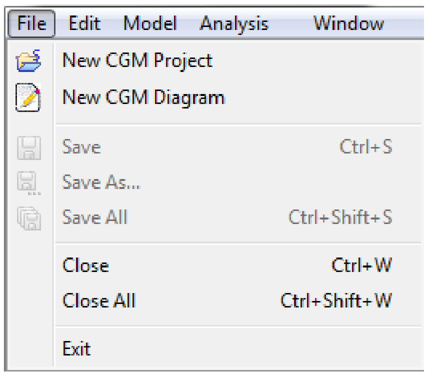


Figure 5: CGM-Tool File menu

(b) Edit Menu:

The modeler can perform **Undo**, **Redo** and **Select All** actions on the active model by accessing commands from this menu(Figure 6).

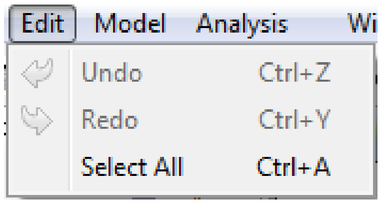


Figure 6: CGM-Tool Edit menu

(c) Model Menu:

The model menu (Figure 7) of CGM-Tool provides basic commands to define **SMT Variables** (see section 4.7 for more) and **Global Constraints** (see section 4.6 for more) into the active goal model.

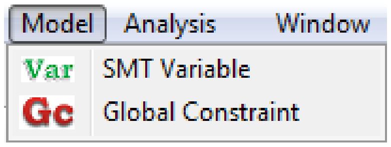


Figure 7: CGM-Tool Model menu

(d) Analysis Menu:

This menu (Figure 8) provide access to the following main functions of the tool:

i. Check Well-formedness:

Performs well-formedness check (see section 5.1) over a CGM model to verify that it comply with the semantics of the modeling language.

ii. Generate Scenario:

This command allows the user to generate a scenario model of the current active model (details about scenario models can be found on section 5.2).

iii. Launch Reasoner:

This command runs *OptiMathSAT* solver over the current active scenario model and presents the reasoning result by coloring the satisfied nodes of the active scenario model. For more information about the reasoning process please see section 5.3 .

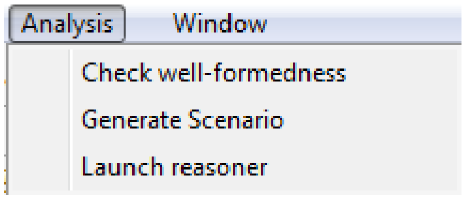


Figure 8: CGM-Tool Analysis Menu

[II] Toolbar

Functions that are commonly performed within the application can be quickly accessed from the tool bar. The tool bar of CGM-Tool accommodates **Check Well-formedness**, **Generate Scenario**, **Launch Reasoner** and few other commands.

2.1.2 Editor Area and Palette

[I] Editor Area

The editor area is the main place where main editing operations (such as creation, deletion, modification...) of the graphical model will be performed. CGM-Tool support opening of multiple editors at the same time.

More details about CGM-Tool editing area and supported operations will be discussed on section 4.

[II] Palette

CGM-Tool palette (Figure 9) supplies a simple drag and drop facility for creation of nodes and relations graphically on the editor area.

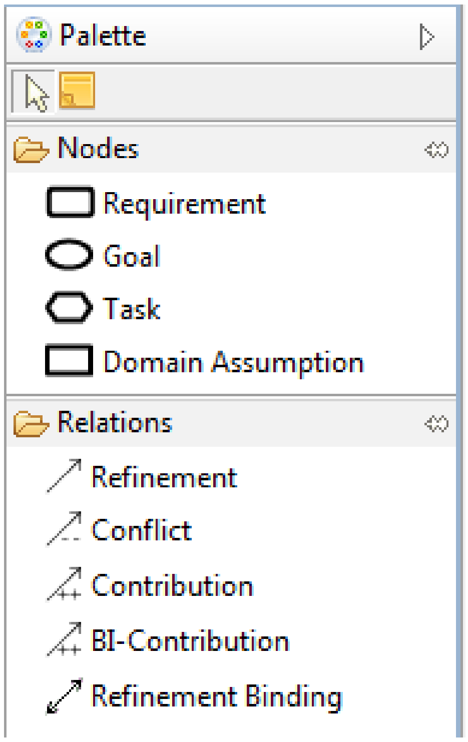


Figure 9: CGM-Tool Palette

Details about the nodes and relations of constrained goal model will be discussed on section 3 and creation of nodes and relation will be covered on section 4.

2.1.3 Views

CGM-Tool allows its users to navigate and modify model data through its views. These views can be freely moved and positioned across the tool GUI. This section discusses some of the main views and basic operations on them, where as other views(such as Property, Analysis Report, Reasoning Result, Node Variable and Model Variable) of the tool will be presented in detail later in this document.

[I] Project Explorer View

It enables to view CGM Projects and its contents hierarchically. Furthermore, it allows modelers to perform the following operation:

(a) Open File

A file in project explorer can be opened by **double clicking** on it. It is also possible to open a file by selecting it with the mouse and **pressing enter on the keyboard**, alternatively you can right click on the file and **select open** from the context menu(Figure 10).

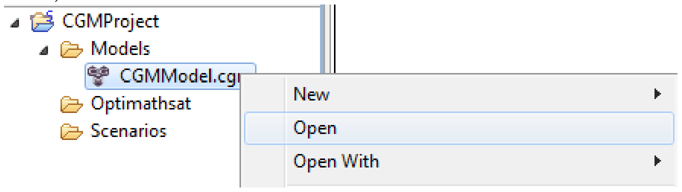


Figure 10: Open file

(b) Rename Project, Folder, and Files

To rename a project, a folder or a file just select it with the mouse, and press **"F2"** on the keyboard, or use the **"Edit menu"** and then select **"Rename"** command. Renaming can be also performed by right-click on the file and select **"Rename in the context menu"**.

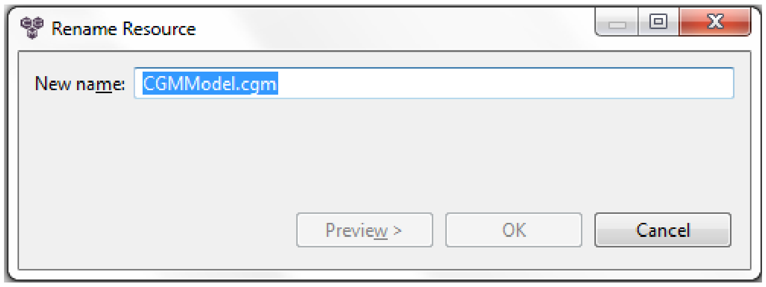
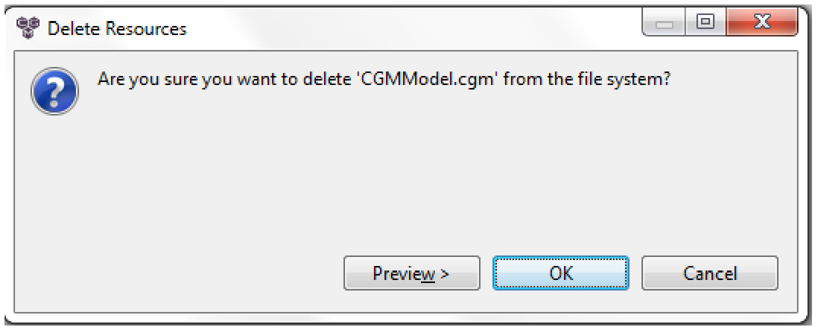


Figure 11: Rename dialog box

The selection of rename command will be followed by a dialog box (Figure 11), that will let you choose the new name. When done press **"Ok"** to confirm or **"Cancel"** to abort the operation.

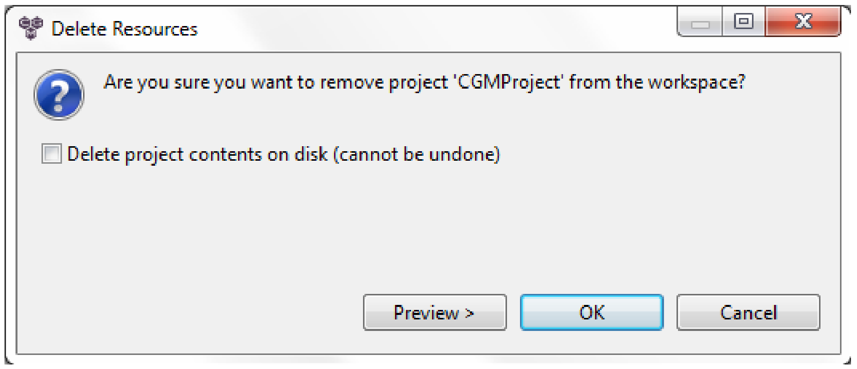
(c) Delete Project, Folder, and Files

To delete a project, a folder or a file left-click on it, and press **"Del"** on the keyboard, or right-click on it and select **"Delete"** command from the context menu.


Delete Resources

Are you sure you want to delete 'CGMModel.cgm' from the file system?

Figure 12: Delete file dialog box


Delete Resources

Are you sure you want to remove project 'CGMProject' from the workspace?

☐ Delete project contents on disk (cannot be undone)

Figure 13: Delete project dialog box

The selection of delete command will be followed by a dialog box(Figure 13), that will require your confirmation for the deletion of the file. The confirmation dialog box is slightly different for project deletion (Figure 12, Which has a check box, that provides option to physically delete the project from the file system. If not checked the project will only be removed from the project explorer view and will be recoverable later, but it will not be possible to create a new project with the same name in the project structure. Select **"Ok"** to confirm the deletion of the file, otherwise press **"Cancel"** to abort the operation.

Note !

- Some default folders in CGM-Tool can not be **Deleted** or **Renamed**!!

[II] Outline View

This view displays the contents of the active editor as list of structural elements. It helps the modeler to easily navigate through the model and to see the overall picture of the model.

The outline view has two parts:

(a) The lower part is a miniature of the entire model.

(b) Where as, the upper part is a hierarchical representation of the model. Selection of an element on this part will automatically reveal the selected element in the editor area.

3 CGM Nodes and Relations

This section presents some of the major constructs (nodes and relations) of constrained goal models. Moreover, it introduces how to use these elements to build a valid goal model, and provides basic modeling guidelines.

3.1 CGM Nodes

In constrained goal models, *Requirements* are used to represent the desired state of affairs that modelers want the system to achieve, these elements can be progressively refined into intermediate goals (*Goal*) until the refinement process produces actionable leaf goals, which require no further decomposition (*Task*). Whereas, propositions about the domain, which need to be held for a goal-refinement relation to work can be represented as *Domain Assumption* nodes in the model. *Refinements* are used to represent the alternative ways to achieve a particular goal in the model. The *refinement* of an element is a conjunction of the sub-elements that are necessary to achieve that element[1].

[I] Requirement
Requirements (Figure 14) are root goals in the model which can be refined into other nodes in the model. However, no other node in the model cannot be refined into it. Graphically it is represented as rounded rectangle, and can be created using "Requirement tool" located in the palette.

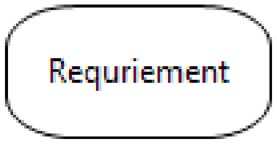


Figure 14: Requirement Node

[II] Goal
Goals (Figure 15) are intermediate goals in a constrained goal model, and can only be refined into other intermediate goal nodes, tasks and domain assumptions in the model. They are graphically represented as Oval, and can be created using "Goal tool" located in the palette.

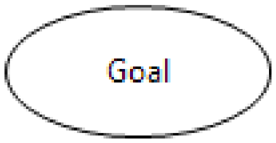


Figure 15: Goal Node

[III] Task

Tasks (Figure 16) are leaf goal in a constrained goal model, and can only be refined into other tasks in the model. Graphically they are represented as “Hexagon”, and can be created using “Task tool” located in the tool palette.

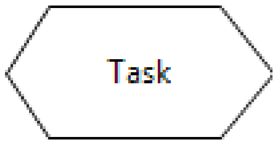


Figure 16: Task Node

[IV] Domain Assumption Node

Domain Assumption (Figure 17) are propositional nodes, and can only be refined into other domain assumption nodes in the model. Graphically Domain Assumption nodes are represented as Rectangle, and can be created using “Domain Assumption tool” located in the CGM-Tool palette.

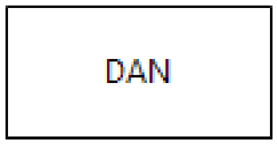


Figure 17: Domain Assumption Node

[V] Refinement

Refinements are represented as a small dark circle, and works as a merging point for the edges connecting a group of source nodes to a target node in a refinement relation. It can be created by using “Refinement tool” located in the CGM-Tool palette.

Note

- If the target of a refinement is a goal node, then its source should contain at least one goal node.
- If the target of a refinement is a domain assumption node, then its source should only contain domain assumptions.
- Refinements should not form cycle.

Table 1 lists available properties for CGM nodes.

Property	Description
Name	Text field which contains name of the selected CGM Node
Description	Text field which contains the general description of the selected CGM Node
Positive constraint	Text field which contains the positive constraint of the node
Negative Constraint	Text field which contains the negative constraint of the node

Table 1: CGM nodes properties

3.2 CGM Relations

CGM Relations/links are graphical ways of specifying constraints in the goal model. CGM-Tool supports expression of the following relation between nodes:

[I] Contribution

If a contribution relation defined between two CGM nodes, it means that satisfying the source node will force the target node to be satisfied. Graphically it is denoted as a directed arrow with "+" label on the middle, and can be created by "Contribution tool" located in the CGM-Tool palette.

[II] Conflict

If a conflict relation defined between two CGM nodes, it means that these two nodes cannot be satisfied simultaneously. Graphically it is denoted as a directed arrow with "-" label on the middle, and can be created by "Conflict tool" located in the CGM-Tool palette.

[III] Bi-Contribution

Bi-contribution relations are used to represent a two side contribution link between two CGM nodes. Graphically it is denoted as a two headed arrow with "++" label on the middle, and can be created by "Bi-Contribution tool" located in the CGM-Tool palette.

[IV] Refinement Binding

These are relations between two refinement nodes. Graphically it is represented as a two headed dashed arrow, and can be created by "Refinement binding tool" located in the CGM-Tool palette.

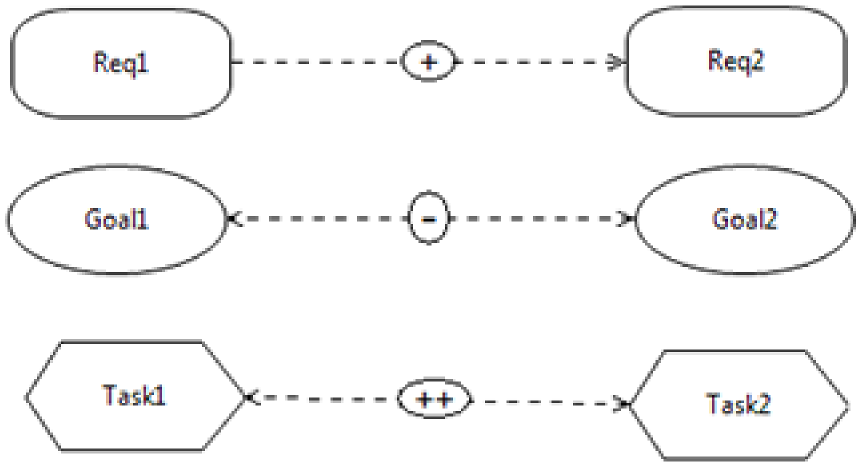


Figure 18: CGM Relations

Note

- All these relations are allowed to form loop in the model.

4 Basic Operations

This section will cover basic operations and techniques that modelers need to follow to build constrained goal models using CGM-Tool.

4.1 Creating New CGM Project

To start modeling and reasoning using CGM-Tool you first need to have at least one active project, thus the tool allows creation of CGM projects in two different ways:

[I] From file menu

Follow the following steps to a create CGM project from the file menu of the tool:

(a) Click on the “File” menu and choose “New CGM Project”.

(b) Give a name for the new project, and click “Finish” button.

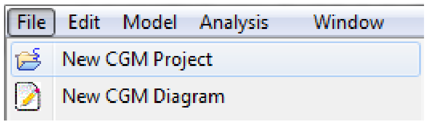


Figure 19: Creating CGM project using File menu

[II] Using project navigator

Follow the following steps to create a CGM project using this option:

(a) Right click anywhere on the “Project Navigator”, then select “New CGM Project” command from “New” context menu.

(b) Give a name for the new project, and click “Finish” button.

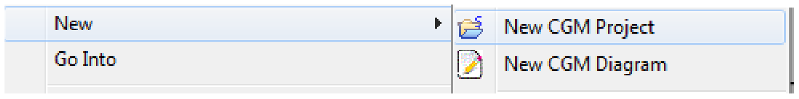


Figure 20: Creating CGM project using Project Navigator context menu

4.2 Creating New CGM Model

Models of constrained goals must always be created inside “Models” folder of a CGM project. And, it can be created either from the “File” menu of the tool or using the context menu of “Project navigator”.

[I] From file menu

  (a) Click on the "File" menu and select "New CGM Model"
  (b) A new model creation wizard will be opened (Figure 21), ensure that the selected parent folder is "Models"
  (c) Give a name for the model and click "Finish" button.

[II] Using project navigator

  (a) Right click anywhere on the "Project navigator", then select "New CGM Model" command from "New" context menu.
  (b) After that, follow step b and c of the above method.

4.3 Adding Nodes

Follow the following steps to add CGM nodes in the editor area (Figure 22:

[I] Select the node that you want to add from the tool palette, by clicking on it.

[II] Move the mouse cursor over the editor canvas (the cursor of the mouse will change its icon this time). Then left click on the area of the editor that you want to add the node.

[III] You can see that the node is added in the required location with its default name. The automatically assigned default name can be change into any other name, by simply type the desired name while the node is active(selected). To return to "selection tool" click outside of element.

Note!

- Names of nodes should be unique in the model.

Tip!

- While creating a node, you can press the "Ctrl" (or "Apple" in case of Macintosh) key on the keyboard to create sequence of nodes with the same type without the need to select the node creation tool repeatedly.

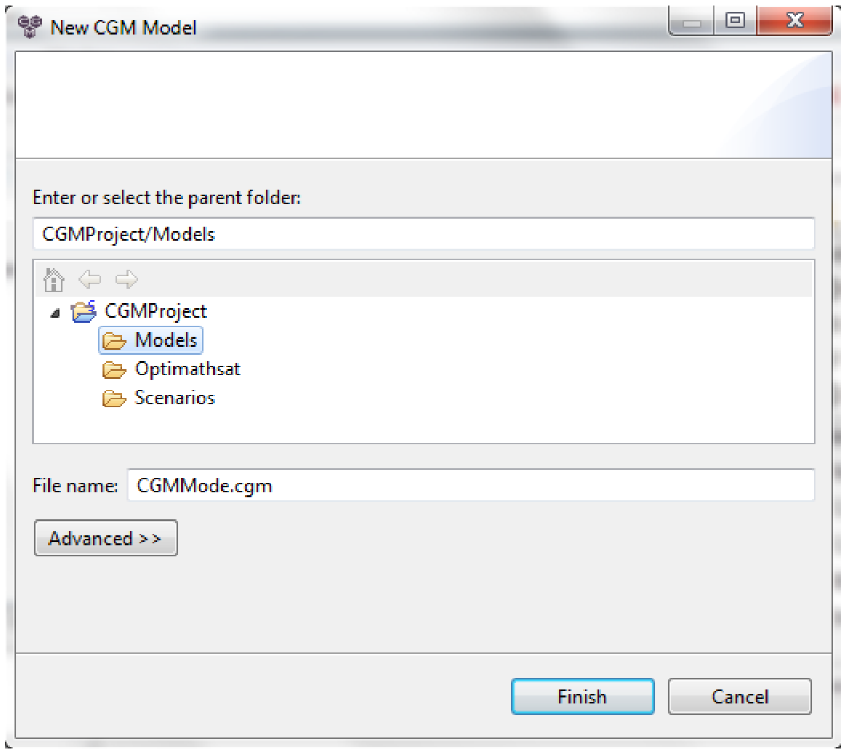


Figure 21: Creating new CGM Model wizard

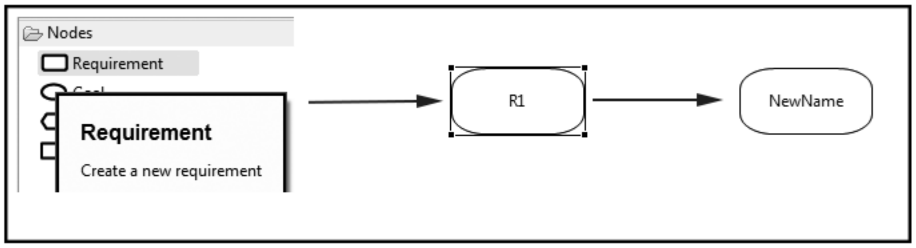


Figure 22: Adding CGM node into the model

4.4 Adding Relations

CGM relations are links between two CGM nodes, as a result in order to create a relation you need to have at least two nodes in the model, which will work as a source and target node of the relation.

The following two steps shows how to create relation between nodes (Figure 23):

[I] Select the required relation from the palette area, and then click on the top of the desired source node.

[II] While keeping the mouse not pressed, move the mouse cursor into the target element, and then click on it. This will create the required relation between the elements.

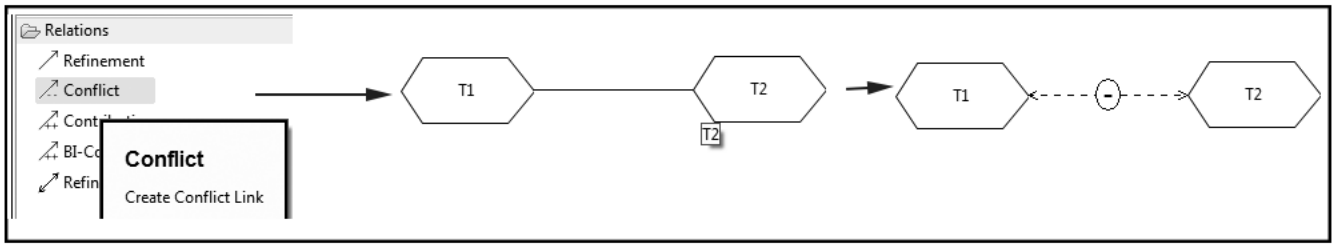


Figure 23: Creating CGM relation

4.5 Force True and Force False Operations

In constrained goal modeling, nodes can be forced either to be true (satisfied), to achieve the desired result, or to be false(denied) to reduce the effort of achieving them.

To force a node to *True* or to *False* (Figure 24), **Right click** on the desired node, and then select either **Force True** or **False Force** command from the context menu. Performing this operation will change the color of the node into *Red* or *Green*, when forced to true or false respectively.

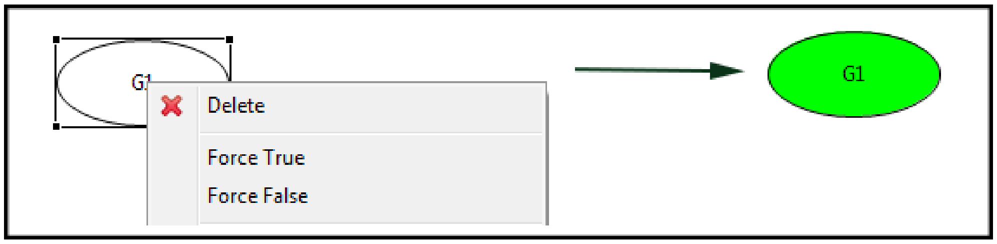


Figure 24: Force node to true/false

4.6 Global and Local Constraints

In addition to constraints which can be expressed graphically in terms of *contribution*, *conflict*, *refinement binding* and *bi-contribution*, modelers can further enrich their models by defining arbitrary prerequisite logic formulas/constraints. These formulas can be expressed either globally to the model or as a local positive and negative constraints for a particular element in the model.

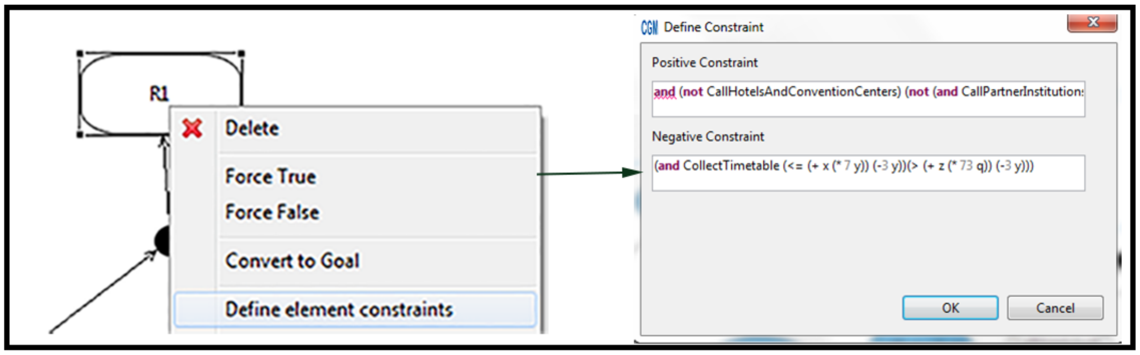
4.6.1 Defining Local Constraints

Local constraints are positive and negative pair of prerequisite formulas expressed on the top of CGM nodes. Where, an element will be satisfied when its positive constraint is satisfied and it will be denied when its negative constraint is satisfied.

To define local constraint for a node follow the following steps (Figure 25):

[I] Right-click on the desired node, and choose "Define element constraint" command from the context menu. A pop-up window with two text fields will appear.

[II] Write the positive and negative constraints/formulas in the text field respectively, and then press OK.



```
and (not CallHotelsAndConventionCenters) (not (and CallPartnerInstitution:
```

```
(and CollectTimetable (<= (+ x (* 7 y)) (-3 y))(> (+ z (* 73 q)) (-3 y)))
```

Figure 25: Defining Element Constraint

Note!

- Constraint formulas must be written based on the SMT BNF listed on Appendix one.

Tip!

- The text field has integrated content assist, pressing **"Ctrl + space"** while writing the formula will reveal the list of available options.

- The syntax-check feature of the constraint text field will assist you to write a syntactically correct SMT formula. It automatically checks the user input, and **underline red** below the incorrect ones.

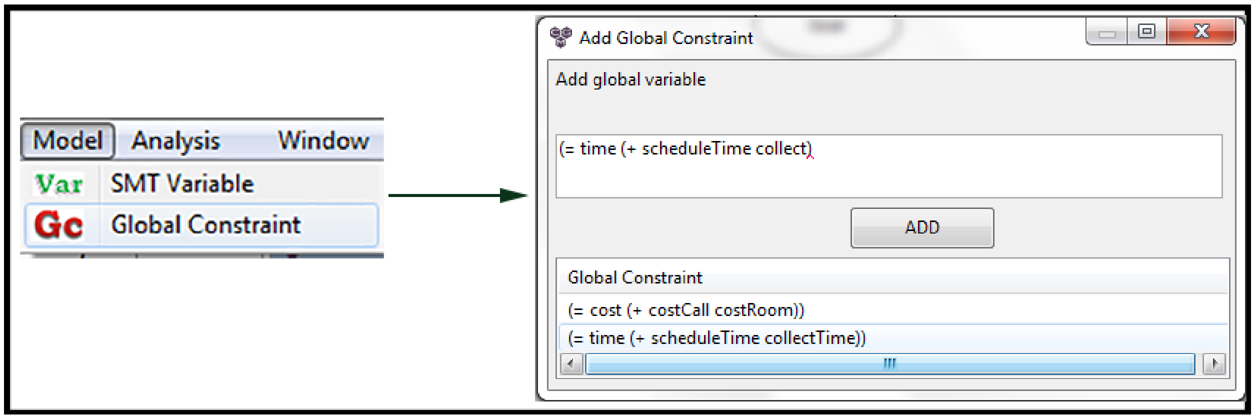
4.6.2 Defining Global Constraints

Global constraints are SMT formulas, which need to be satisfied for the whole model to be satisfied. These formulas can be defined on CGM models by using the following steps (Figure 26):

[I] From the tool bar (or from Model menu in the menu bar) select "Global Constraint" command, it will open a pop-up window.

[II] In the pop-up window "Add global constraint" text field type the formula of the constraint you want to define, and press "Add".

[III] Global formulas can be many, therefore follow the above step until you finish defining the constraints. In case if you want to remove the previously defined constraint form the list, you can do so by **right-click on the formula** and select **"Remove constraint"**, in addition double clicking on the top of a formula will let you to **modify/update** the formula.



(= time (+ scheduleTime collect)

(= cost (+ costCall costRoom))
(= time (+ scheduleTime collectTime))

Figure 26: Add Global Constraint

4.7 SMT Variables

SMT variables in CGM are global numeric variables used to express different numerical attributes of elements (such as cost, work time, space, fuel, etc...)

4.7.1 Variable Declaration

Follow the following steps to define SMT variables to the underlying model:

[I] From the tool bar (or from the "Model menu" of the menu bar) select "SMT Variable" command, as a result a new pop up window will be opened.

[II] Write the name of the variable in the "variable name" text field and choose "Define" Button

Property	Default value	Availability	Description
Positive value	0	Node-level	
Negative value	0	Node-level	
Upper bound	0	Model-level	The upper bound of the value of the variable
Lower bound	0	Model-level	The lower bound of the value of the variable
Priority	0	Model-level	The priority order of the variables :Lowest number (i.e. 0) has highest priority
Objective	MINIMIZE	Model-level	Stakeholder's objectives towards the value of the variable in terms of minimizing or maximizing.
Calculation method	Default	Model-level	The choice of stakeholders for computation of the variable outcome. Stakeholders can either follow the default calculation method of the tool or define a calculation method by themselves.

Table 2: List of SMT variables attributes

[III] It is possible to define more than one variable to the model, for this follow the above step repeatedly until you finish defining any number of variables. In case, if you want to **remove** a previously defined variable from the list, you can do so by **right-click** on the required variable from the list and select **"Remove variable"** context menu.

Note!

- **weight** is a default variable for all constrained goal models,therefore it is not possible to remove/change it from SMT variable list.

SMT variables are composed of several model-level and node-level fields/attributes. Node level fields are available for all element nodes except intermediate goals (i.e. goals) in the model. Table 2 presents the summary of these attributes, their default values and availability (in terms of location where you can define their values).

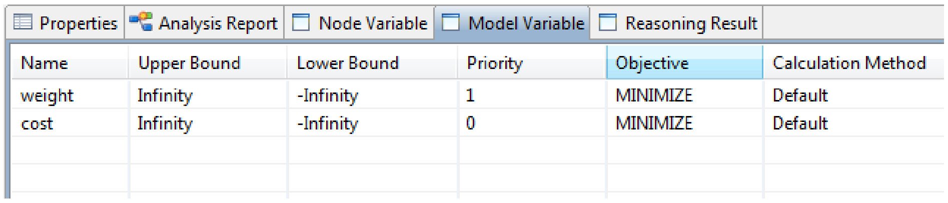
4.7.2 Variable Naming

[I] SMT variable name can be any legal identifier - which begin with a letter and followed by any combination of letters and digits.

[II] SMT variable names are **case sensitive**, therefore please make sure that you wrote the exact variable name while utilizing them inside prerequisite constraints.

4.7.3 Setting Model Level SMT Variable Values

You can set/modify model level (global) values of SMT variables (such as *Upper bound*, *Lower bound*, *Priority*, *Objective* and *Calculation Method*) by going into **"Model variable"** table (Figure 27), located below the editor area of the tool.

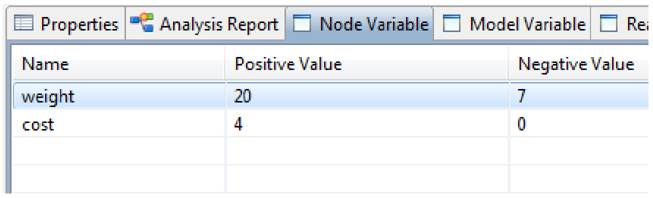


Name	Upper Bound	Lower Bound	Priority	Objective	Calculation Method
weight	Infinity	-Infinity	1	MINIMIZE	Default
cost	Infinity	-Infinity	0	MINIMIZE	Default

Figure 27: Model SMT Variables Table

4.7.4 Setting Node Level SMT Variable Values

Node level SMT variable values such as positive and negative values can be expressed in **"Node Variable"** table, which is located right next to **"Model variable"** table. This table is filled with list of variables and their values for the actively selected node on the editor.



Name	Positive Value	Negative Value
weight	20	7
cost	4	0

Figure 28: Node SMT Variables Table

4.8 Frequent Modeling Operations

4.8.1 Converting Nodes to Each Other

CGM-Tool enables modelers to automatically convert nodes in the model to one another. Perform the following steps to access these functionality (Figure 29):

[I] **Right click** on the node that you want to convert.

[II] From the list of context menus (i.e.“**Convert to Requirement**”, **“Convert to Task”**, **“Convert to Goal”** and **“Convert to DAN”**) choose the target node type into which you want to covert.

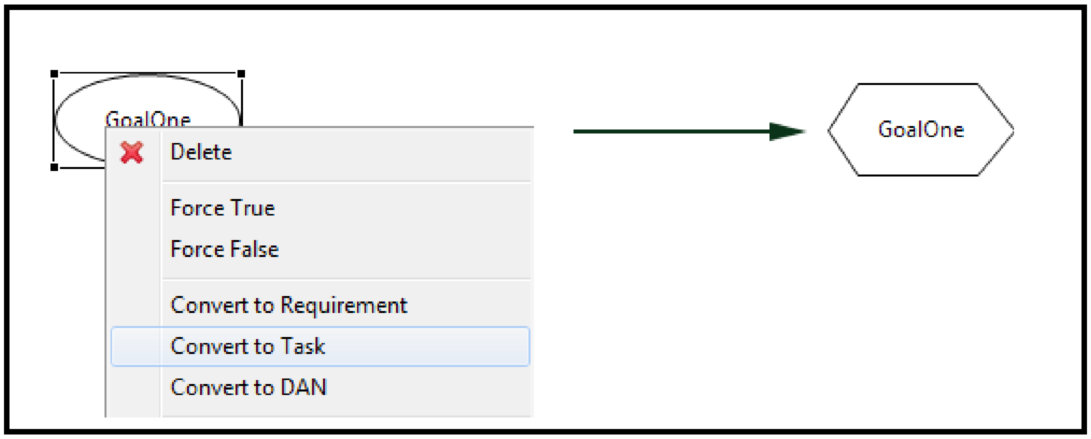


Figure 29: Converting Node

Note!

- In order to maintain the validity and correctness of the model, the availability of these node conversion commands will be customized based on the refinement relations of the selected node. Table 3 presents the availability requirements of each function on a particular node.

- All node conversion commands are available on **Isolated CGM Nodes**. Isolated nodes are neither source nor target node of any refinement relation in the model.

4.8.2 Converting Relations to Each Other

In addition to the support for converting nodes, CGM-Tool allows modelers to convert relations to each other. Perform the following steps to access these functionality.

[I] **Right click** on the relation that you want to convert.

[II] From the list of context menus (i.e.“**Convert to Contribution**”, **“Convert to Conflict”** and **“Convert to Bi-Contribution”** choose the target relation type into which you want to covert.

4.8.3 Elements Deletion

Use on of the following ways to delete CGM elements from the model:

[I] From keyboard: Select the Node/Relation you want to delete and press **“Delete”** key from your keyboard.

[II] By Mouse: **Right click** on the Node/Relation you want to delete, and then select **“Delete”** from the pop-up menu (Figure 30).

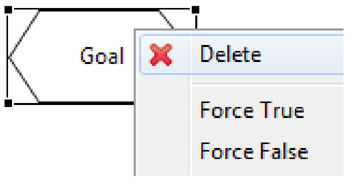


Figure 30: Deleting CGM Element

4.8.4 Renaming Nodes

CGM nodes can be renamed by the following way:

- Select the node that you want to change its name, and type the new name of the node, press enter to confirm.

Note!

- Node names in the model should be unique.

4.8.5 Moving Elements and Relations

[I] Moving Nodes:

Nodes can be moved by selecting them, and dragging them around to a new position. While dragging a node on the model editor area a phantom figure will show the new location-to-be of the node (Figure 31).

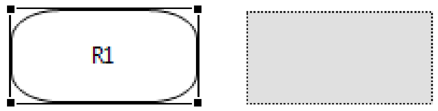


Figure 31: Moving CGM Node

[II] Moving Relations

To move relations you first need to select them. When selected, a relation will display a little handle on the middle of the link figure (Figure 32), this handle can be dragged with a mouse to the new location you want to move.

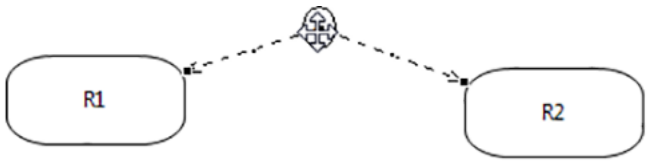


Figure 32: Moving CGM Relation

4.8.6 Undo/Redo Operations

All operations performed on CGM-Tool editor can be undone or redone using support of the tool. You can perform undo/redo operation using one of the following ways:

[I] Keyboard shortcut:

- Ctrl (Command for Mac) + Z for Undo
- Ctrl (Command for Mac) + Y for Redo

[II] Menu bar:

- The the **"Edit"** menu of the tool contain "Undo" and "Redo" commands for the respective operations.

[III] Tool bar:

- From the tool bar of the tool, choose the undo/redo icons for the undoing or redoing operations.

Node conversion command	Availability requirement
Convert to Requirement	• Available only if the selected node is **Root node** (i.e. A node in the model into which no other node is refined).
Convert to Goal	• Available only if the selected node is **Leaf node** (i.e. A node which is not refined into any other node in the model).
Convert to Task	• Available only if the selected node is **Leaf node** (i.e. A node which is not refined into any other node in the model).
Convert to DAN	• Available only if the selected node is **Leaf node** (i.e. A node which is not refined into any other node in the model).

Table 3: Node conversion commands and their availability requirement

5 Well-formedness Analysis, Scenario Generation and Reasoning

Beside the support for modeling constrained goals, CGM-Tool is equipped with other functionalities that helps to perform well-formedness check and automated reasoning on the underlying model, furthermore it enables generation of reasoning report in pdf format.

5.1 Well-formedness Check

The well-formedness check of CGM-Tool helps to examine and verify the well-formedness and validity of constrained goal models. A CGM is considered to be well-formed if its constitute elements (nodes and relations) are modeled and interconnected according to the syntax and semantics of the modeling language. Thus, the provided functionality for well-formedness check performs the following post-modeling analysis to verify the compliance of the underlying model with the semantics of the modeling language.

[I] **Empty diagram check:** verify the emptiness of the given model. If it is empty none of the succeeding well-formedness checks will be performed. However, if it has contents(i.e. nodes and relations), the check will return *"No error found"* and will perform the next check.

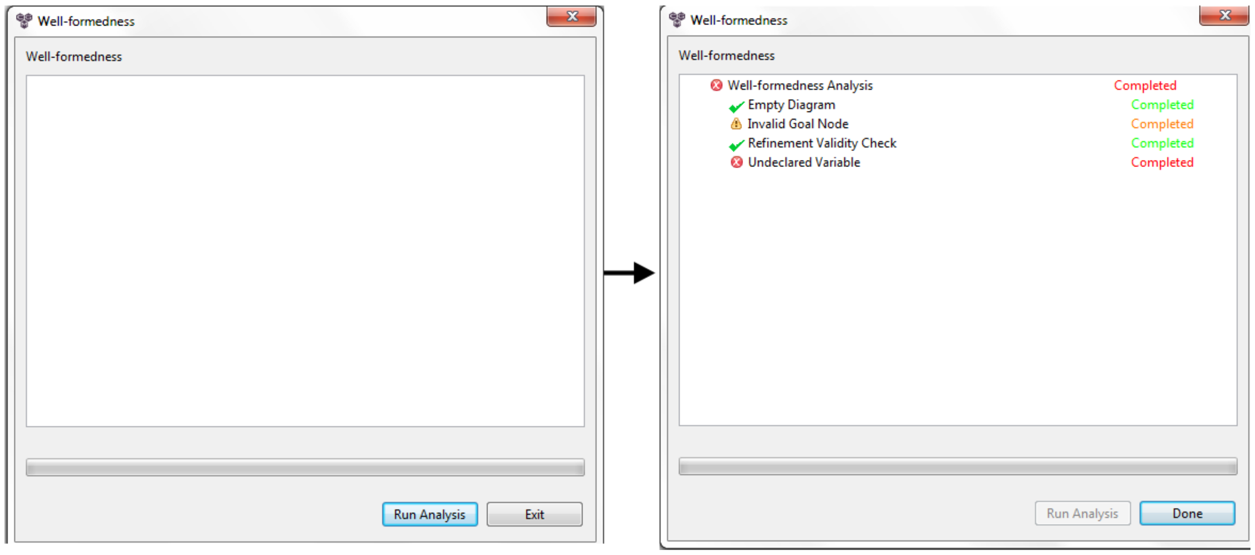
[II] **Invalid goal node check:** This check examines the validity of goal nodes in the model. *According to CGM, every root goals should be* ***Requirement*** *and leaf goals should be* ***Task****.* This check helps to identify goal nodes that violates this demand of the modeling language.

[III] **Invalid refinement check :** Based on the modeling language of CGM,*if the target of a refinement relation is goal node, then at least one of the sources of that refinement should be a goal node.* This check helps to verify whether the underlying model satisfies this requirement or not. If the model contain nodes that breach this need, the check will automatically return their lists and possible causes.

[IV] **Undeclared variables check:** *Modelers must declare every numeric variables that they are using in both global and local constraints.* The purpose of this check is to examine every user defined constraints in the model, and identify the use of any undeclared SMT variables inside it. And in case of discovery, it will notify the name of the undeclared variable and its location (i.e. name of specific node or global constraint) where it is being used.

5.1.1 Launch Well-formedness Analysis

Well-formedness analysis can be launched through **"Check well-formedness"** command of **"Analysis menu"** , Alternatively you can also launch it through **"Well-formedness check"** icon of the tool bar. Once you choose the well-formedness check command, a dialog box with an empty white area will appear (Figure 33). To start the analysis you must press **"Run Analysis"** button. This will automatically start the analysis engine, subsequently tasks being performed and their completion status will fill the white area one by one, while the progress of the analysis also shown on the progress bar of the dialog.



```
Well-formedness Analysis          Completed
    Empty Diagram                 Completed
    Invalid Goal Node             Completed
    Refinement Validity Check     Completed
    Undeclared Variable           Completed
```

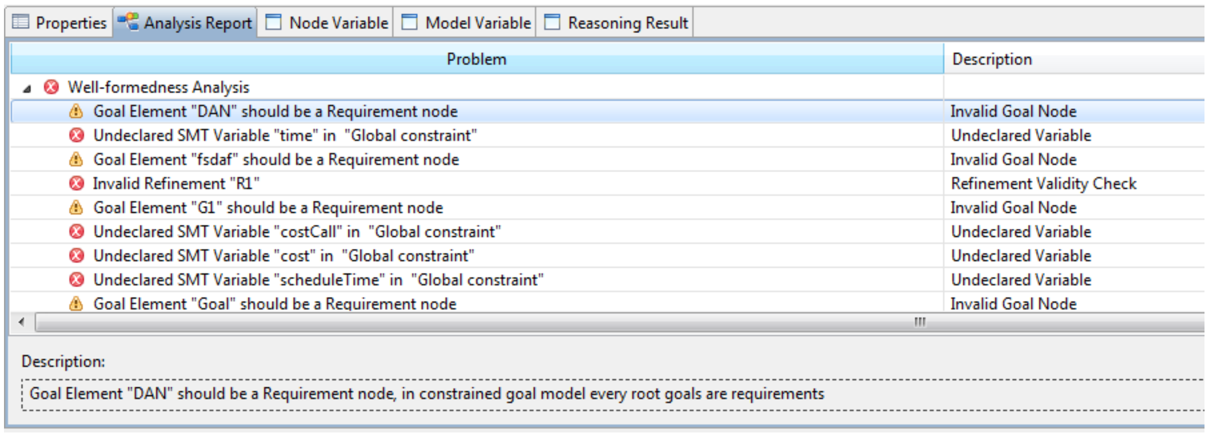
Figure 33: Well-formedness analysis dialog

Completion status of each checks of the analysis can be categorized into one of the following results:

- **Started**: The task has started and is performing the required check.
- **Completed (Green)**: The task completed without finding any errors.
- **Completed (Orange)**: The task completed, and found some non-serious errors (warnings).
- **Completed (Red)**: The task completed, and found some serious errors

- **Exception** : The task has encountered an internal error while executing. (This should never occur, in case you revive this error, please contact the CGM-Tool support).

Upon completion, lists of errors and warnings discovered in the model by the analysis will be displayed in the **"Analysis Report"** table (Figure 34) of the tool.



Problem	Description
Well-formedness Analysis	
Goal Element "DAN" should be a Requirement node	Invalid Goal Node
Undeclared SMT Variable "time" in "Global constraint"	Undeclared Variable
Goal Element "fsdaf" should be a Requirement node	Invalid Goal Node
Invalid Refinement "R1"	Refinement Validity Check
Goal Element "G1" should be a Requirement node	Invalid Goal Node
Undeclared SMT Variable "costCall" in "Global constraint"	Undeclared Variable
Undeclared SMT Variable "cost" in "Global constraint"	Undeclared Variable
Undeclared SMT Variable "scheduleTime" in "Global constraint"	Undeclared Variable
Goal Element "Goal" should be a Requirement node	Invalid Goal Node

Description:
Goal Element "DAN" should be a Requirement node, in constrained goal model every root goals are requirements

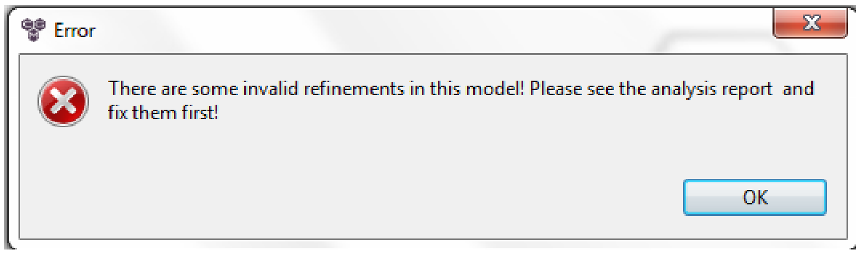
Figure 34: Analysis report table

5.2 Scenario Generation

Scenario models are well-formed and valid copies of the original model, their purpose is to provide a possibility to model a small scenario of a very complex system, and to perform reasoning on it. It enhances the reasoning process by providing the copy of the model for the user to play with (by changing and modifying the scenario model until the desired result achieved) while keeping the original model unmodified.

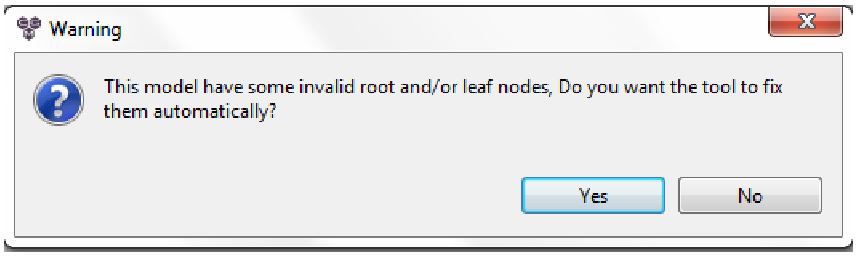
5.2.1 Generate Scenario Model

The scenario generation wizard can be launched by choosing **"Generate scenario"** command from **"Analysis menu"**, alternatively it can also be launched by **"Generate scenario"** icon of the tool bar. Once you choose "Generate scenario" command, an automatic well-formedness check will be performed on the model, and if the model contain severe errors (such as *invalid refinement node* and/or *undeclared SMT variable*) it will notify the modeler to correct these problems (Figure 35) and will halt the scenario generation process. However, if the automatic well-formedness analysis identify only less severe error/warnings (such as *invalid goal node*) it will report the warnings for the modeler and will give an option to automatically correct these problem by itself (Figure 36). If everything goes fine and if no problem detected by the analysis a dialog box with a text field for the name of the scenario will appear on the screen (Figure 37), give it a valid name and press **"Finish"** to complete the scenario generation process.


Error

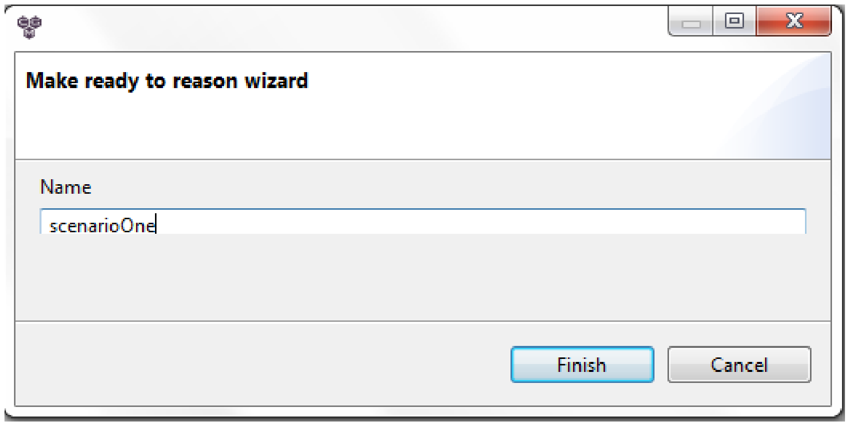
There are some invalid refinements in this model! Please see the analysis report and fix them first!

Figure 35: Scenario generation error notification


Warning

This model have some invalid root and/or leaf nodes, Do you want the tool to fix them automatically?

Figure 36: Scenario generation warning notification


Make ready to reason wizard

Name: scenarioOne

Figure 37: Scenario generation dialog box

The generated scenario model will be stored inside **"Scenarios"** folder of the current project. Specifically it will be found inside a folder named by the original model name, which reside inside **"Scenarios"** folder (Figure 38).

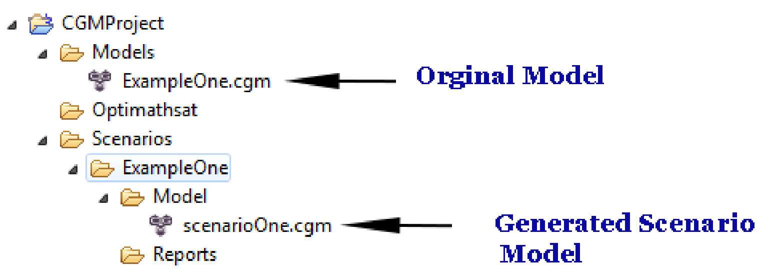


Figure 38: Generated scenario location

5.3 Reasoning on the Model

CGM-Tool provides automated mulit-objective reasoning with unsat-core extraction. This feature enables modelers to reason on the model, and examine whether the underlying model is realizable or not. To provide this functionality CGM-Tool utilized **OptiMathSAT SMT-Solver**.

5.3.1 Launch Reasoner

To launch reasoner wizard on a CGM model, select **"Launch Reasoner"** command from **"Analysis menu"**, alternatively it can be launched by **"Launch Reasoner"** icon of the toolbar. Launching reasoner command will automatically initiate well-formedness check on the model. If the check does not discover any problem, the tool will encode the graphical model into SMT2 formula, then it will call OptiMathSAT solver on the top of it. After that, the reasoning result file, which is generated by OptiMathSAT solver will be parsed. If the model is realizable/satisfied then the results will be presented in the following format:

- Highlight parts of the model (in blue) which are realizable/satisfied (Figure 39).
- Generate a pdf report for the reasoning result, which contains the **diagrammatic result**, **encoded smt2 file** and the **OptiMathSAT output** file.
- And, tabular result (Figure 40), that is filled with the calculated values of all non-zero SMT variables and the total reasoning time.

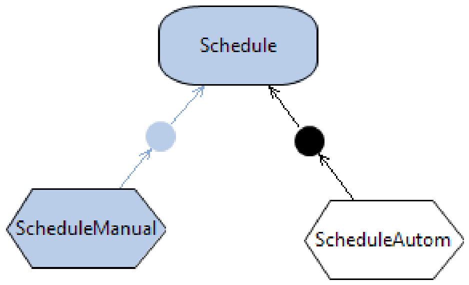
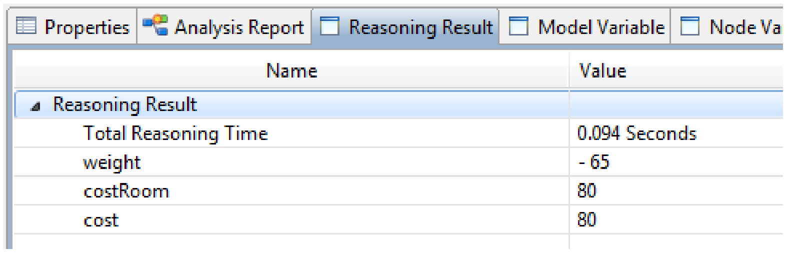


Figure 39: Highlighted Satisfied Nodes

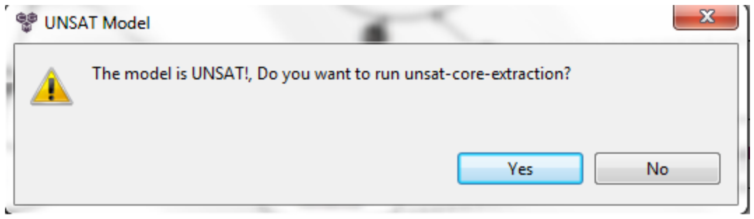


Name	Value
Reasoning Result	
Total Reasoning Time	0.094 Seconds
weight	- 65
costRoom	80
cost	80

Figure 40: Reasoning Result Table

However, if the solver identifies that the model is unrealizable/unsatisfied the tool will give you an option to reveal parts of the model which made the whole model unrealizable (Figure 41). Proceeding with this option will run unsat-core extraction on the model and produce the results for the unachievability of the model in the following format.

- Highlight parts of the model in yellow, which are unrealizable/unsatisfied (Figure 42).
- Generate a pdf report for unsat-core extraction result, which contains the **diagrammatic result**, **encoded smt2 file** and the **OptiMathSAT output** file.
- And, tabular result (Figure 43),filled lists of encountered problems that made the model unsatifiable.


UNSAT Model

The model is UNSAT!, Do you want to run unsat-core-extraction?

Figure 41: UnSAT core extraction dialog box result

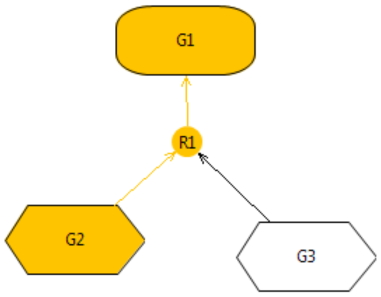
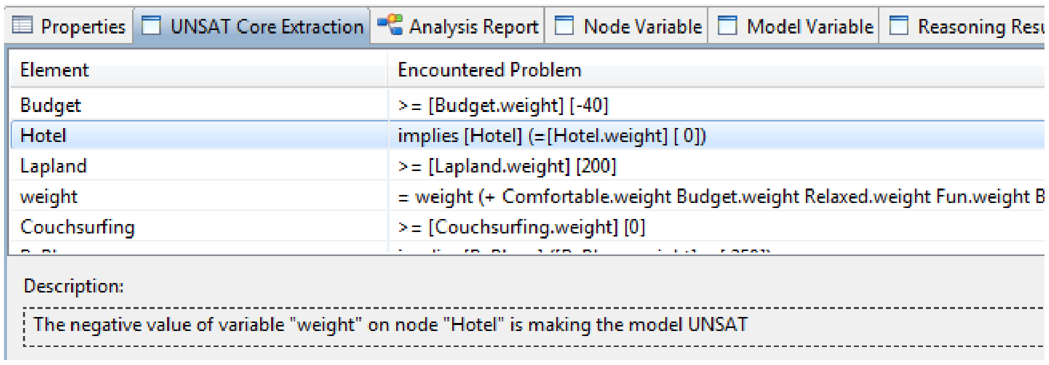


Figure 42: Highlighted UnSAT Core result



Element	Encountered Problem
Budget	>= [Budget.weight] [-40]
Hotel	implies [Hotel] (=[Hotel.weight] [0])
Lapland	>= [Lapland.weight] [200]
weight	= weight (+ Comfortable.weight Budget.weight Relaxed.weight Fun.weight B
Couchsurfing	>= [Couchsurfing.weight] [0]

Description:

The negative value of variable "weight" on node "Hotel" is making the model UNSAT

Figure 43: UnSAT Core extraction tabular result

During the reasoning process the encoded formula and the OptiMathSAT output file will be stored in the **Optimathsat folder** of the current project. And, the reasoning report will be found on the **"Report"** folder of the active Scenario folder.

6 Acknowledgment

The research leading to these results has received funding from European Research Council (ERC) under grant number 267856 (Lucretius).

Appendices

A BNF of SMT Formulas

```
⟨FORMULA⟩     ::= ⟨ATOM⟩
                | (and ⟨FORMULA⟩[⟨FORMULA⟩]+)
                | (or ⟨FORMULA⟩[⟨FORMULA⟩]+)
                | (not ⟨FORMULA⟩)
                | (xor ⟨FORMULA⟩ ⟨FORMULA⟩)
                  // Note: in SMT language ‘‘iff’’ is written as ‘‘=’’
                | (= ⟨FORMULA⟩ ⟨FORMULA⟩)
                | (implies ⟨FORMULA⟩ ⟨FORMULA⟩)
                  // Syntactic-sugaring formulas
                  // if-then-else
                | (Ite ⟨FORMULA⟩ ⟨FORMULA⟩ ⟨FORMULA⟩)
                  // Equivalent to xor
                | (Alternative ⟨FORMULA⟩ ⟨FORMULA⟩)
                  // Equivalent to implies
                | (Causes ⟨FORMULA⟩ ⟨FORMULA⟩)
                | (Requires ⟨FORMULA⟩ ⟨FORMULA⟩)
                  // Cardnality boolean constraints
                | (OneOf ⟨FORMULA⟩[⟨FORMULA⟩]+)
                | (AtMostOneOf ⟨FORMULA⟩[⟨FORMULA⟩]+)
                | (AtLeastOneOf ⟨FORMULA⟩[⟨FORMULA⟩]+)
                | (ExactlyNOf ⟨INTEGER-NUMBER⟩ ⟨FORMULA⟩[⟨FORMULA⟩]+)
                | (AtMostNOf ⟨INTEGER-NUMBER⟩ ⟨FORMULA⟩[⟨FORMULA⟩]+)
                | (AtLeastNOf ⟨INTEGER-NUMBER⟩ ⟨FORMULA⟩[⟨FORMULA⟩]+)

                  // NOTE: for boolean formulas only (no SMT)the preceding op-
                  tion is sufficient

⟨ATOM⟩        ::= ⟨BOOLEAN-ATOM⟩
                | ⟨MATH-ATOM⟩

⟨BOOLEAN-ATOM⟩ ::= ⟨IDENTIFIER⟩
```

```
⟨IDENTIFIER⟩ ::= [a-zA-Z][a-zA-Z0-9]+

⟨MATH-ATOM⟩ ::= (⟨OP⟩ ⟨MATH-TERM⟩ ⟨MATH-TERM⟩)

⟨OP⟩          ::= =
                |  >
                |  >=
                |  <
                |  <=

⟨MATH-TERM⟩ ::= ⟨MONOMIAL⟩
                |  (+ ⟨MONOMIAL⟩ [⟨MONOMIAL⟩]+)
                |  (- ⟨MONOMIAL⟩ ⟨MONOMIAL⟩)

⟨MONOMIAL⟩  ::= ⟨NUMBER⟩
                |  ⟨VARIABLE⟩
                |  (* ⟨NUMBER⟩⟨VARIABLE⟩)

⟨NUMBER⟩     ::= ⟨INTEGER-NUMBER⟩
                |  ⟨REAL-NUMBER⟩

⟨INTEGER-NUMBER⟩ ::= [-][0-9]+

⟨REAL-NUMBER⟩ ::= ⟨INTEGER-NUMBER⟩
                |  [-][0-9]*.[0-9]+

⟨VARIABLE⟩   ::= ⟨IDENTIFIER⟩
```

7 Acronym

CGM : Constrained Goal Model

DAN : Domain Assumption Node

ERC : European Research Council

GUI : Graphical User Interface

OS : Operating System

RCP : Rich Client Platform/Application

SMT : Satisfiability Modulo Theories

References

[1] Mai Chi Nguyen, Roberto Sebastiani, Paolo Giorgini, and John Mylopoulos. Multi-objective reasoning with constrained goal models.

[2] Roberto Sebastiani and Patrick Trentin. Optimathsat: A tool for optimization modulo theories.