DesignSpaceHelper: Used to Apply Constraints on a Design Space and Manage Generated Configurations

Overview

The DesignSpaceHelper analyzes and applies the constraints defined in the CyPhyML design space and generates the configurations that satisfy those constraints. This component can be used to export the selected/all configurations into in a *Configurations* model within the top design container. Each exported configuration is represented as a *Configuration With Constraints* (CWC) model in the container. A CWC model stores the references of the selected components in the configuration and references to the constraints that were applied to arrive at these configurations. Additionally, it contains a reference to the Design Space for which the configurations were generated. These are useful aids in traceability of a generated configuration.

It is important that the CWC models contain only references to actual models in the design space without any connections or hierarchy. This is because, for any realistic design space, there are thousands of possible configurations at the static analysis stage. Using fully elaborated models can quickly deplete all available system memory. Another tool, called CyPhyCAExporter, can be used to fully elaborate chosen configurations (CWC models). A fully elaborated ComponentAssembly will include all of the connections between components and subsystems and preserves the hierarchy of the design space.

Allowable constraints

The following constraints are allowed in the design space: Contextual Non-linear OCL constraints, Visual constraints, Parameter constraints, DecisionGroup constraints, Property constraints, and Conditional Property constraints.

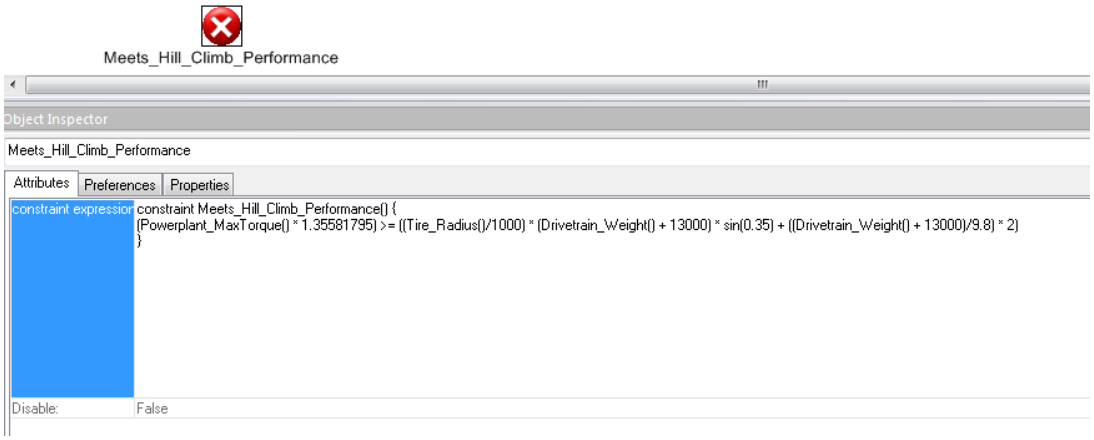

Figure 1(a): Context constraint: Meets_Hill_Climb_Performance

Contextual Non-linear OCL constraints: These are written textually in OCL format and are associated with the container it contains in the context with which it must be satisfied. *Figure 1(a)* depicts an example of the Context constraint. This ensures that the IFV drivetrain is capable enough to accelerate

on a 20-degree uphill at an acceleration of 2 m/s^2. Additionally, the language supports the use of trigonometric functions.

Also, Table 1(a) lists all the mathematical functions that are supported for the OCL constraints.

S. No.	Mathematical Function	Description
1.	*sin*	SINE of a given number in radians
2.	*cos*	COSINE of a given number in radians
3.	*tan*	TANGENT of a given number in radians
4.	*asin*	INVERSE SINE of a given number in radians
5.	*acos*	INVERSE COSINE of a given number in radians
6.	*atan*	INVERSE TANGENT of a given number in radians
7.	*sinh*	HYPERBOLIC SINE of a given number in radians
8.	*cosh*	HYPERBOLIC COSINE of a given number in radians
9.	*tanh*	HYPERBOLIC TANGENT of a given number in radians
10.	*asinh*	INVERST HYPERBOLIC SINE of a given number in radians
11.	*acosh*	INVERSE HYPERBOLIC COSINE of a given number in radians
12.	*atanh*	INVERSE HYPERBOLIC TANGENT of a given number in radians
13.	*log2*	LOGARITHM TO THE BASE 2 of a given number
14.	*log10*	LOGARITHM TO THE BASE 10 of a given number
15.	*ln*	NATURAL LOGARITHM (i.e., to the base 'e') of a given number
16.	*exp*	EXPONENTIAL FUNCTION (i.e., e^x)
17.	*sqrt*	SQUARE ROOT of a given number
18.	*sign*	SIGNUM FUNCTION (returns -1 for –ve, 0 for 0, and +1 for +ve number)
19.	*rint*	MATHEMATICAL ROUND FUNCTION returning double value
20.	*abs*	ABSOLUTE VALUE FUNCTION

Table 1(a): Mathematical functions supported in DESERT

Visual constraints: These are helpful in specifying compatibility constraints. It allows to group DesignElement references in *AND* & *OR* groups connecting them with implied relationships. For example, it can be used to specify that Non-C7 Engines are only compatible with larger transmission CX31 (*Figure 1(b)*).

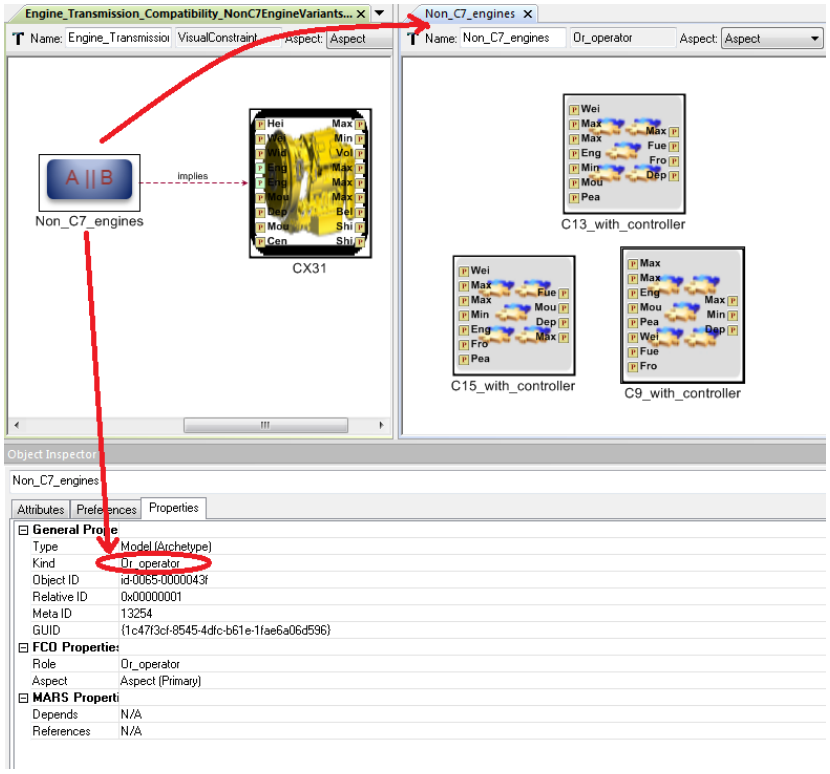

Figure 1(b): Visual constraint: Non-C7-Engines compatibility constraint with transmissions

Parameter constraints: These are constraints that get automatically generated at runtime depending on the parametric ranges specified by the user for values of certain parameters of design elements. For example, Figure 1(d) below shows that the 'sprint_constant' parameter was specified by the user to have a value between 0.75 and 45. The default value was given as 45. This leads to the creation of a parameter constraints automatically that forces the value of the parameter 'spring_constant' to satisfy the valid parametric range provided by the user.

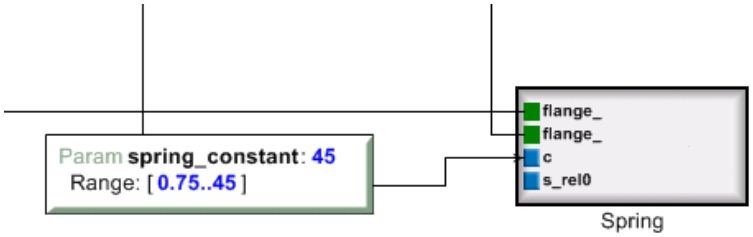

Figure 1(c): Parameter constraint generation as a result of parametric specification of 'spring_constant' parameter

DecisionGroup constraints: These are a good utility available for the users to be able to succinctly specify a set of compatibility constraints for selection of design elements. For example, if the system design contains the use of 4 tires and each tire has a choice from 3 different types, the user can specify a set of Visual constraints that make sure that if a particular tire type is chosen for one of the 4 tire choices in a configuration, then the other 3 tires must also be of the same type. However, this is not sufficient

because it must be specified for all available tire types. Not only that user also needs to specify this constraints in reverse directions (i.e., both A → B and B → A). As such, it quickly becomes an arduous task of specifying several visual constraints manually. To ease this, DesignSpaceHelper provides an easy way to specify such constraints where choice (decision) made for one of the alternative design container must also be chosen for all other design containers present in a DecisionGroup parent model. For example, Figure 1(d) below specifies that "Same tire type (viz. A, B, or C) is chosen for all 4 tire choices (viz. TireFrontLeft, TireFrontRight, TireRearLeft, TireRearRight).

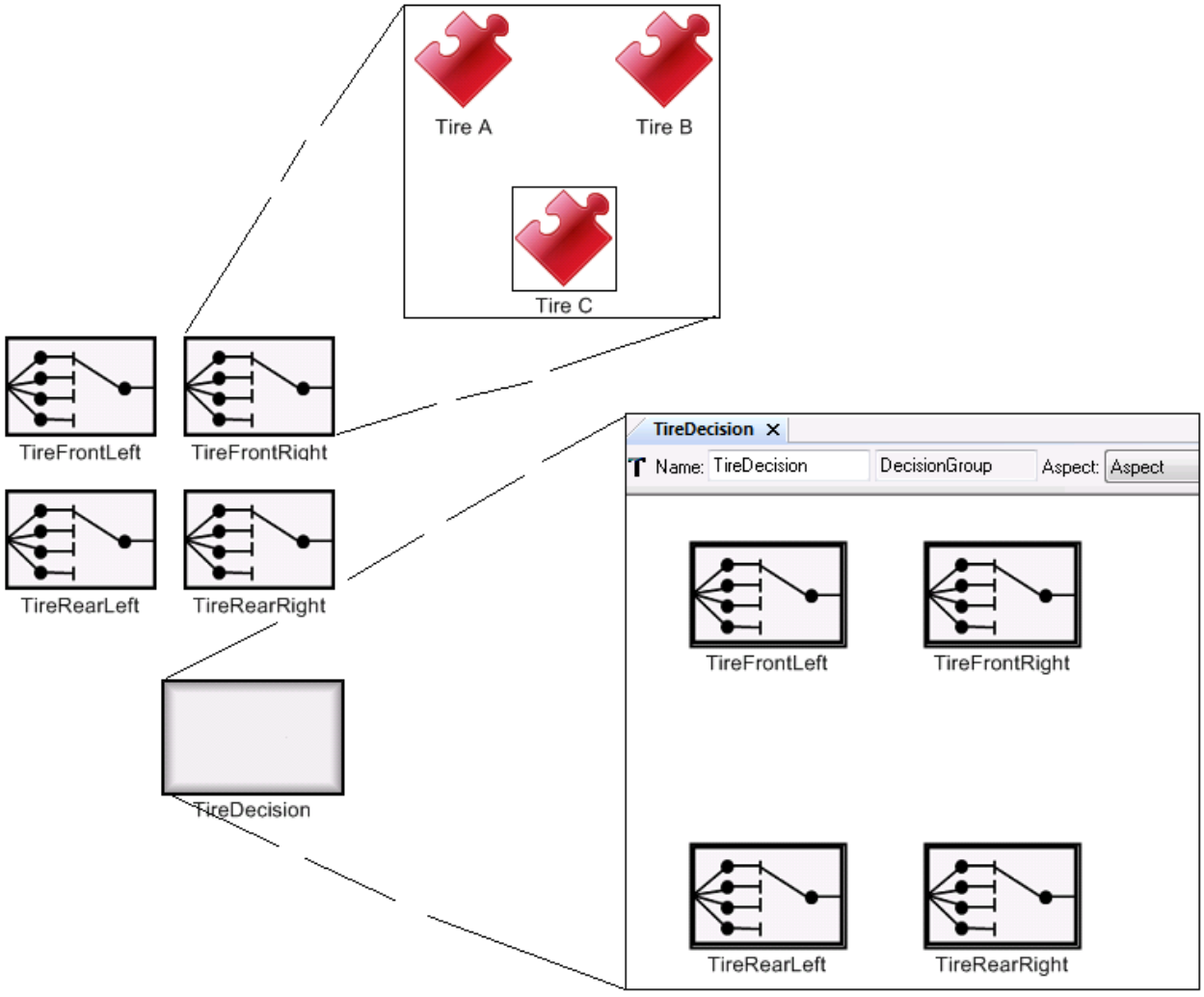

Figure 1(d): Compatibility among set of similar choices using DecisionGroupConstraints

Property constraints: These are a new addition in the CyPhy language. Given a target value for a property, a property constraint allows a user to specify if the value for that property in any of the generated configurations should be less than, less than or equal to, equal to, greater than or equal to, or greater than the target value. *Figure 1(e)* shows an example property constraint.

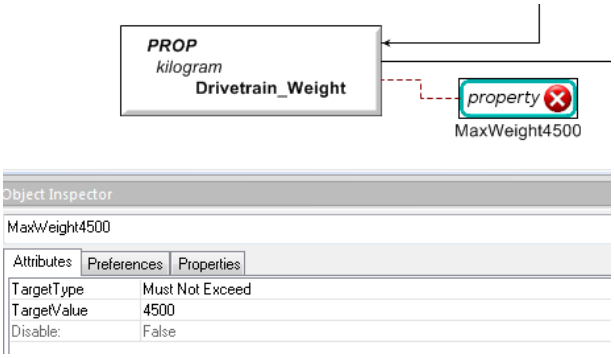

Figure 1(e): Property constraint: Overall Drivetrain weight must be less than or equal to 4500 Kg

Conditional Property constraints: These are similar to Property constraints but are conditioned on a given condition. This condition is specified by a VisualConstraint using the AND/OR operators and Implies connections as shown in Figure 1(b). The condition may represent existence of a group of components or any arbitrarily complex constraint that must hold for the Conditional Property constraint to be valid. Once the Property constraint and Visual constraint for the condition are modeled, the two can be tied by adding a reference of the Property constraint inside the Visual constraint. This transforms the Property constraint into a Conditional Property constraint as shown in figure 1(f) below.

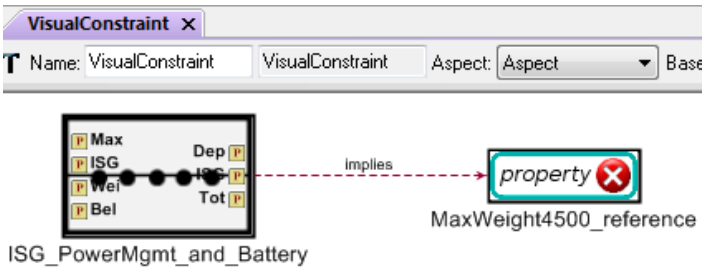

Figure 1(f): Example of a Conditional Property Constraint

In above example, the same Property constraint that was described in Figure 1(e) is now conditioned on the selection of an ISG and Battery assembly in the configuration. As such, the property constraint will be applied only for those configurations where the ISG and Battery assembly was selected.

Usage

CyPhyML DesignSpace Constraints GUI:

Figure 2(a) below shows the main Design Space Exploration Tool GUI. This dialog is shown when the Design Space Helper component is invoked on a design space. If a design space is not currently in focus in GME, the component will show a dialog to select from the design space choices that are available in the model.

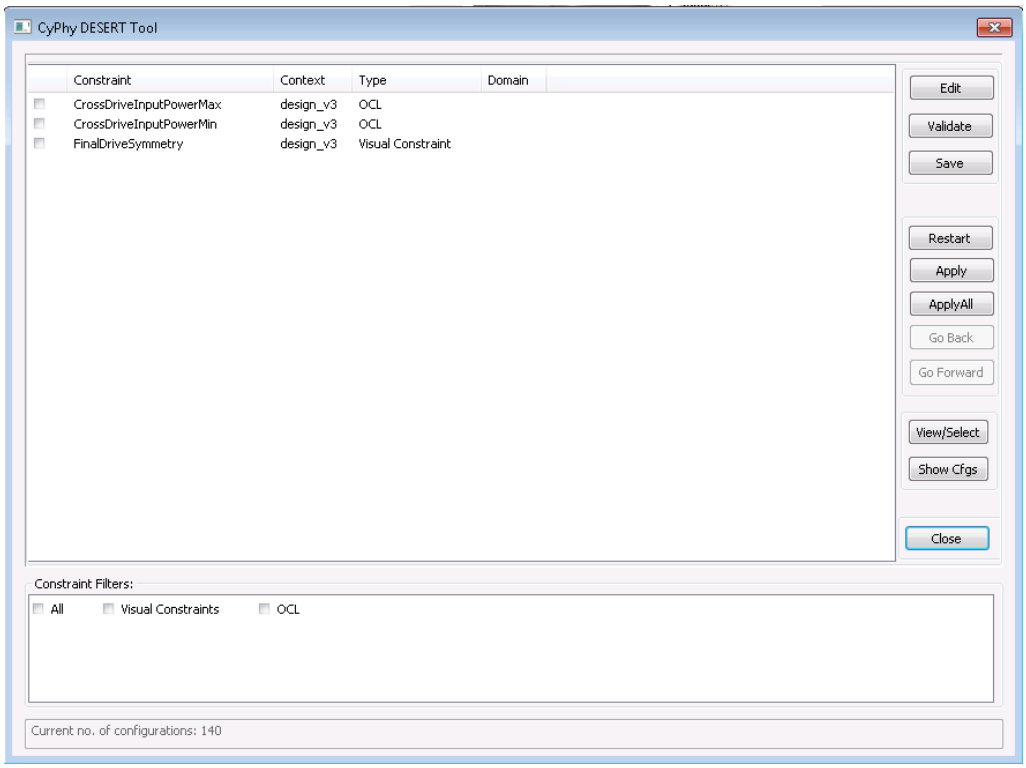

Figure 2(a): DESERT: Design Space Exploration Tool

As shown above, the main panel shows the list of constraints defined in design space model.

- ***Edit:*** Edit the expression of the checked constraint. Valid function list is provided.
- ***Validate:*** Check the validity of all the constraints. If there is any error, the error information will be shown.
- ***Save:*** Save all the changes of the constraints back into model.
- ***Restart:*** Goes to initial state when no constraints were applied.
- ***Apply:*** Apply the checked constraint(s) to design space.
- ***ApplyAll:*** Apply all constraints to design space.
- ***View/Select:*** This can be used to down-select certain components. This is useful when we know that we must certain components for some of the choices in the design space. This can greatly reduce number of configurations that needs to evaluated.
- ***Show Cfgs:*** The generated configurations will show up.
- ***Close:*** Exit the DesignSpaceHelper tool.

The constraints can be applied incrementally. "***Go back***" and "***Go Forward***" can be used to navigate back or forward between design space results from applied constraints.

Furthermore, the bottom panel shows various constraint filters available for the constraints listed in the top panel. In the models, users can specify a constraint domain for all constraints. Additionally, the constraints have their types as one of the domains already listed. As such, a constraint can have more than one domain. Using these domains, the constraints are grouped and the filters are provided so that user can choose to selectively apply constraints belonging to the domains of interest.

Generated Configurations GUI:

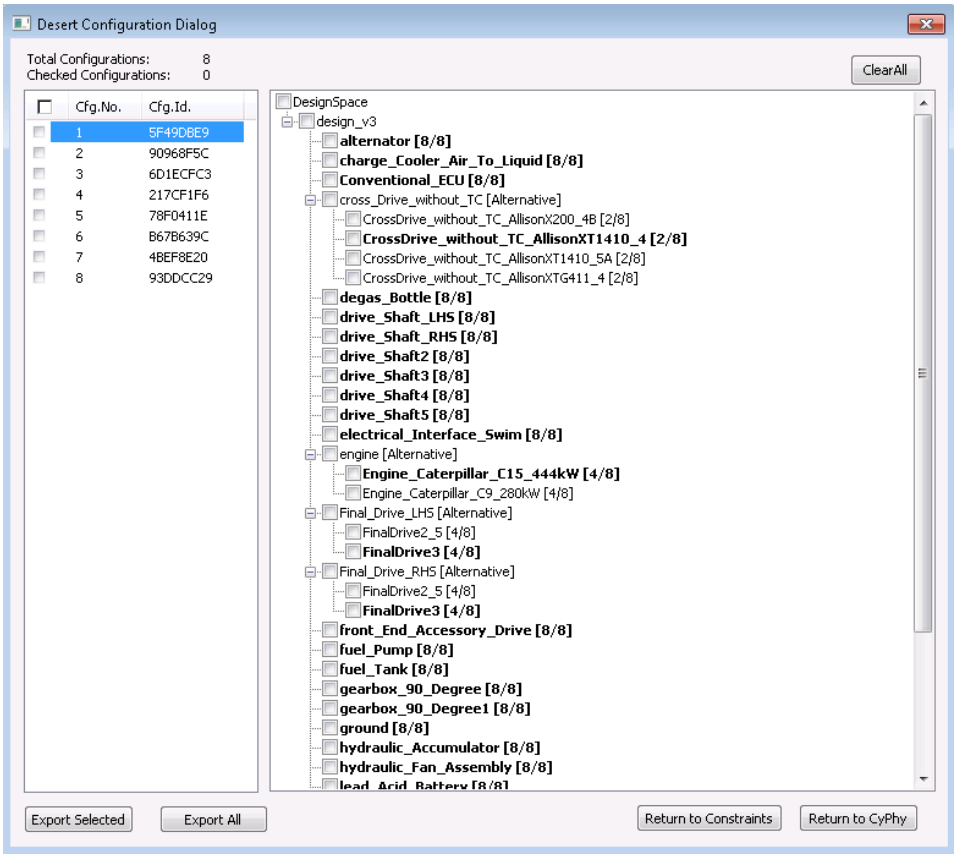

Figure 2(b): The Design Space Configurations list after apply constraints

As shown above, the left panel shows the list of generated configurations by applying the constraints from previous step. The right panel shows the hierarchical (tree) view of the design space.

A user can click on the configuration and see all selected components highlighted. For example, in the above figure 2(b), the configuration 1 is selected. As soon as this configuration is selected, the components that form this configuration in the design space tree are highlighted in bold face.

Moreover, this dialog also allows users to select key options that they want in all of the exported configurations. Checking appropriate checkboxes in the right hand side design tree completes this step. When options are selected, the appropriate configurations are automatically selected in the left side panel. The default behavior is to select only those configurations (intersection) that include all of the chosen options in the design tree. However, when multiple design options of a single Alternative container are chosen, all configurations that include any of the selected options are selected (union).

The dialog also contains several command buttons, which work as follows:

Exported Selected: Export the selected configuration(s) into CWC model(s).

Exported All: Export the all configurations into CWC models.

Return to Constraints: Goes back to the prior constraint dialog.

Return to CyPhy: Closes all dialogs and goes back to GME.