ASP Benchmark generator for Configuration Problem

user guide – version 1.0

Piotr Gorczyca
gorczycapj@gmail.com

September 2020

Contents

1 General information **3**

2 Encoding **3**

2.1 Taxonomy . . . 3

2.1.1 Creating a taxonomy . . . 3

2.1.2 Adding a component . . . 4

2.1.2.1 Adding a component's sibling . . . 4

2.1.2.2 Adding a component's child . . . 4

2.1.3 Renaming a component . . . 4

2.1.4 Removing a component . . . 4

2.1.4.1 Remove component preserving its children . . . 4

2.1.4.2 Remove component and all its children . . . 4

2.2 Associations . . . 4

2.2.1 Adding an association . . . 4

2.2.1.1 Setting lower bound . . . 4

2.2.1.2 Setting upper bound . . . 5

2.2.1.3 Setting exact number . . . 5

2.2.2 Editting an association . . . 5

2.2.3 Removing an association . . . 5

2.3 Ports . . . 5

2.3.1 Adding a port type . . . 5

2.3.2 Renaming a port type . . . 5

2.3.3 Removing a port type . . . 5

2.3.4 Forcing connection of a port type . . . 5

2.3.5 Setting compatibility of a port type . . . 6

2.3.6 Creating component's port individuals . . . 6

2.4 Resources . . . 6

2.4.1 Adding a resource . . . 6

2.4.2 Renaming a resource . . . 6

2.4.3 Removing a resource . . . 6

2.4.4 Setting component's resource production/consumption . . . 6

2.5 Constraints . . . 7

2.5.1 Simple constraints . . . 7

2.5.1.1 Adding a simple constraint . . . 7

2.5.1.2 Editing a simple constraint . . . 7

2.5.1.3 Simple constraint window . . . 7

2.5.1.4 Removing a simple constraint . . . 7

2.5.2 Complex constraints . . . 8

2.5.2.1 Adding a complex constraint . . . 8

2.5.2.2 Editing a complex constraint . . . 8

2.5.2.3 Complex constraint window . . . 8

2.5.2.4 Removing a complex constraint . . . 9

3 Instances **9**

3.1 Setting an exact number of instances . . . 9

3.2 Setting a bounded number of instances . . . 9

3.3 Symmetry breaking . . . 9

3.3.1 Setting symmetry breaking for a component . . . 10

3.3.2 Setting symmetry breaking for all components . . . 10

4 Generation **10**

5 Solving **10**
5.1 Generation and solving 11

1 General information

This guide explains how to use the **benchmark generator** to create the ASP encodings of configuration problem instances, solve them and extract answers. The program utilizes the ideas presented in [1] such as taxonomy, associations, ports, resources and constraints. The concept of symmetry breaking and variable number of component's type instances is based on [2].

2 Encoding

Modelling of the configuration problem with the benchmark generator is divided into two parts, following the good practices of ASP modelling – **encoding** and **instances**. The first part allows user to define the rules of a correct configuration, whereas the latter one is used (mostly) to define the actual number of components available. All encoding-related options are available in the **Encoding** tab, located in the top-left corner of the program's window.

2.1 Taxonomy

Components form a tree stucture, a taxonomy of elements – from the most abstract to the most tangible ones. **Leaf** components (viz. components with no children) correspond to a real world object, whereas **non-leaf** components always form some kind of a "category". An example a non-leaf component can be an *operating system*, while *Windows 10 Home 64-bit* would be its leaf (grand)child.

There is only one **root component** – the component that's configuration is the goal of this program.

In order to modify the taxonomy select the **Taxonomy** tab in the left panel.

2.1.1 Creating a taxonomy

To create a complete taxonomy press the button **Create taxonomy**, which is located at the bottom of the **Taxonomy** tab. You will be presented with a text area, where you can enter the desired taxonomy. Each line must contain only one component. If component A, located above component B is on the same indentation level as A, that means the components are **siblings** in the taxonomy tree. If B is supposed to be A's child, it should be preceeded with tabulation. E.g.

```
componentA
        componentAs_child
componentB
        componentBs_child
                componentBs_childs_child
```

Warning: If you select **Create taxonomy** option after taxonomy has already been created, even if you don't change much in the taxonomy you will lose all information about constraints and associations, information about which ports the components have and the amount of resources they produce. If you want to make subtle changes only, consider using options on the right of the **Taxonomy** tab (such as **Rename component**, **Add sibling** etc).

2.1.2 Adding a component

2.1.2.1 Adding a component's sibling

To add a sibling, select the component that you want to add the sibling of (from the treeview on the left-hand side), then press **Add sibling** button from the right panel.

2.1.2.2 Adding a component's child

To add a child, select the component that you want to add the child of (from the treeview on the left-hand side), then press **Add child** button from the right panel.

2.1.3 Renaming a component

To rename component, select the component you want to rename (from the treeview on the left-hand side), then press **Rename component** button from the right panel.

2.1.4 Removing a component

2.1.4.1 Remove component preserving its children

To remove a component, but preserve its children, select the component that you want to remove (from the treeview on the left-hand side), then press **Remove** button from the right panel. Selected component will be removed, but its children will become the children of the parent of the removed component.

2.1.4.2 Remove component and all its children

To remove a component and all its children, select the component that you want to remove, then press **Remove recursively** button from the right panel.

Note: When removing a component that is referenced in any **constraint**, the constraint referencing it will be removed as well. If that happens, you will be notified which constraints have been removed.

2.2 Associations

In the **Associations** tab you can configure the one-to-many associations between the root component and other components. E.g. you can state that a valid *pc* must contain precisely 1 *power supply unit* and between 1 and 6 *disk drives*.

2.2.1 Adding an association

To add an association, select a component (from the treeview on the right-hand side), then tick the **Has association?** checkbox located on the left-hand side, right under the selected component's name. This will enable the **Has min?** and **Has max?** checkboxes.

2.2.1.1 Setting lower bound

To set the association's lower bound, tick the **Has min?** checkbox. That will enable the entry under the checkbox, which can be used to enter the desired lower bound value. To remove the lower bound, simply untick the **Has min?** checkbox.

2.2.1.2 Setting upper bound

To set the association's upper bound, tick the **Has max?** checkbox. That will enable the entry under the checkbox, which can be used to enter the desired upper bound value. If you want to restrict that the configuration must not contain a component of some type, input 0 to the upper bound entry. To remove the upper bound, simply untick the **Has max?** checkbox.

2.2.1.3 Setting exact number

To set the association's exact number, tick both **Has min?** and **Has max?** checkboxes, then input the same number to both entries under the checkboxes.

2.2.2 Editting an association

To edit an association, select the component (from the treeview on the right-hand side) that you want to edit the association of, then follow the instructions in the section above in order to modify the association according to your needs.

2.2.3 Removing an association

To remove the association from the component, select the component (from the treeview on the right-hand side) that you want to remove the association from, then simply untick the **Has association?** checkbox.

2.3 Ports

Ports tab is used to model connections or compatibility between individual components. A port type is a definition of a connection interface. It contains the information about the **compatibility** of the port type – set of other port types whose port individuals can be connected to individuals of the port type.

2.3.1 Adding a port type

To add a port type press **Add** button from the left panel.

2.3.2 Renaming a port type

To rename a port type select the port type that you want to rename (from the combobox on the top of the left panel), then press **Rename** button from the left panel.

2.3.3 Removing a port type

To remove a port type select the port type that you want to remove (from the combobox on the top of the left panel), then press **Remove** button from the left panel.

2.3.4 Forcing connection of a port type

To force the connection of a port type select the port type that you want to force connection of (from the combobox on the top of the left panel), then tick the **Force connection** checkbox from the left panel.

Forcing connection of a port type enforces that a component containing the port cannot appear in the configuration, unless the port is connected. E.g. assume we have components that are: a *motherboard* and a *graphics card*. The *motherboard* contains many ports and not all of them are required to be connected (similarly, we do not have to connect all USB, HDMI and jack ports so

that we can use our PCs). On the other hand, the *graphics card* port, used to connect it to the *motherboard* has to be connected, otherwise the *graphics card* would be useless. In such situations the **Force connection** option may deem useful.

2.3.5 Setting compatibility of a port type

To set the compatibility a port type select the port type that you want to set the compatibility of (from the combobox on the top of the left panel), then press the **Edit compatibility** button from the left panel. You will be presented with a new window, containing two lists of ports. To add another port type to the set of compatible port types select it from the **Port** list (located on the left-hand side) and press the >> button. To remove a port type from the list of compatible port types, select it from the **Compatible ports** list (located on the right-hand side) and press the << button.

2.3.6 Creating component's port individuals

To set the number of port individuals a component should have select the port type (from the combobox on the top of the left panel), then select a component (from the treeview on the left-hand side). Next enter the amount of port individuals in the **Has:/Children have:** entry. If the selected component is a leaf component, then amount will be updated automatically. If the selected component is not a leaf component however, you can press the **Apply to all children** button to apply the amount of port individuals of the port type to all leaf (grand)children of the selected component.

2.4 Resources

The concept of a **resource** is useful whenever some components produce some kind of entity, whereas others consume it. The amount produced must always be greater or equal to the amount consumed, so that the balance is always greater or equal than 0. For example, *disk space* may be a resource produced by *disk drives* and consumed by *software*.

In order to modify the resources and set components' resource production/consumption, select the **Resources** tab in the left panel.

2.4.1 Adding a resource

To create a new resource press **Add** button from the left panel.

2.4.2 Renaming a resource

To rename a resource, select the resource that you want to rename (from the combobox on the top of the left panel), then press **Rename** button from the left panel.

2.4.3 Removing a resource

To remove a resource, select the resource that you want to rename from the combobox on the top of the left panel, then press **Remove** button from the left panel.

2.4.4 Setting component's resource production/consumption

To set the component's resource **production**, select the resource that you want to set the production of (from the combobox on the top of the left panel), then select the component (from the treeview on the right-hand side). Next enter the produced amount in the **Produces:** entry. If the selected component is a leaf component, then amount will be updated automatically. If the selected component is not a leaf component however, you can press the **Apply to all children** button to apply the production amount to all leaf (grand)children of the selected component.

To set the component's resource **consumption** the only difference is to enter a negative amount in the **Produces:** entry.

2.5 Constraints

Constraints tab is used to define conditions a correct configuration must satisfy.

2.5.1 Simple constraints

Simple constraints are similar to **Associations**, but are more expressive and have different meaning. They are used to enforce a number (or range) of components of some type present in the configuration.

2.5.1.1 Adding a simple constraint

To add a simple constraint press the **Add simple constraint** button (from the right panel). You will be presented with the **Simple constraint window**.

2.5.1.2 Editing a simple constraint

To edit a simple constraint, select a simple constraint that you want to edit (from the **Constraints** list on the left-hand side), then press the **Edit** button (from the right panel). You will be presented with the **Simple constraint window**.

2.5.1.3 Simple constraint window

- To **add the components** that are required by the constraint, select component from the **Component** treeview (located on the left-hand side) and press >> button to add the selected component only or >> **(recursively)** to add the selected component and all its (grand)children.
- To **remove the component** from the set of component required by the constraint select this component from the **Selected components** treeview (located on the right-hand side) and press << button.
- To **set the number of instances of selected components the configuration must contain**, tick the **Has min?/Has max?** checkboxes and input the lower/upper bound respectively. If an exact number is desired, then lower and upper bounds should be equal. To ensure that a configuration does not contain any instances of selected component, input 0 in the max entry.
- To **set whether instances of one component class should be counted once only**, tick the **Distinct?** checkbox. If not, every instance of will be counted individually.
- To **set the constraint's name** input the name in the **Name:** entry (located on the top of the right-hand side).
- To **set the constraint's description**, input the description in the **Description:** entry (located on the top of the right-hand side, under the **Name:** entry). Description is optional, but may be useful to write down constraint's meaning (especially when the constraint is complicated).

2.5.1.4 Removing a simple constraint

To remove a simple constraint, select a simple constraint that you want to remove (from the **Constraints** list on the left-hand side), then press the **Remove** button (from the right panel).

2.5.2 Complex constraints

Complex constraints are more powerful than their simple counterparts. They can enforce a conditional constraint – if some condition is satisfied, then some other condition must be satisfied too in order to keep the configuration correct. E.g. if a *pc* has more than 4 *disk drives*, then a powerful *power supply unit* is required in order to supply the pc with enough power.

2.5.2.1 Adding a complex constraint

To add a simple constraint press the **Add simple constraint** button (from the right panel). You will be presented with the **Complex constraint window**.

2.5.2.2 Editing a complex constraint

To edit a complex constraint, select a complex constraint that you want to edit (from the **Constraints** list on the left-hand side), then press the **Edit** button (from the right panel). You will be presented with the **Complex constraint window**.

2.5.2.3 Complex constraint window

- Condition
  - To **add a part of condition** press the **Add** button (located in the left-bottom of the window, under the **Condition** list). You will be presented with **Simple constraint window** which is used to create the part of the complex constraint's condition's part.
  - To **edit a part of condition**, select the part of condition that you want to edit (from the **Condition** list), then press the **Edit** button (located under the list you selected the part of condition from). You will be presented with **Simple constraint window** which is used to edit the part of the complex constraint's condition's part.
  - To **remove a part of condition**, select the part of condition that you want to remove (from the **Condition** list), then press the **Remove** button (located under the list you selected the part of condition from).
  - To set if **all** condition parts must be satisfied in order to satisfy the condition (and activate the constraint), tick the **all** radiobutton (located under the **Condition** list. If satisfying **any** of the condition parts is enough to safisfy the condition (and activate the constraint), tick the **any** radiobutton.
- Consequence
  - To **add a part of consequence** press the **Add** button (located in the right-bottom of the window, under the **Consequence** list). You will be presented with **Simple constraint window** which is used to create the part of the complex constraint's consequence's part.
  - To **edit a part of consequence**, select the part of consequence that you want to edit (from the **Consequence** list), then press the **Edit** button (located under the list you selected the part of consequence from). You will be presented with **Simple constraint window** which is used to edit the part of the complex constraint's consequence's part.
  - To **remove a part of consequence**, select the part of consequence that you want to remove (from the **Consequence** list), then press the **Remove** button (located under the list you selected the part of consequence from).
  - To set if **all** consequence's parts are to be enforced (provided that the condition is satisfied), tick the **all** radiobutton (located under the **Condition** list). If enforcing **any** of the consequence parts is enough, tick the **any** radiobutton.

- To **set the constraint's name** input the name in the **Name:** entry (located on the top of the left-hand side).
- To **set the constraint's description** input the description in the **Description:** entry (located on the top of the left-hand side, under the **Name:** entry). Description is optional, but may be useful to write down constraint's meaning (especially when the constraint is complicated).

2.5.2.4 Removing a complex constraint

To remove a complex constraint, select a complex constraint that you want to remove (from the **Constraints** list on the left-hand side), then press the **Remove** button (from the right panel).

3 Instances

The **instances** tab serves the purpose of setting the number of instances of component types available. It can be selected from the top-left corner of the program window.

3.1 Setting an exact number of instances

To set an exact number of instances of a component type select the component type from the component treeview on the right-hand side. Next, on the left-hand side, under the selected component's name, tick the **Exact** radiobutton and input the number of entries in the **Count:** entry below. If the selected component is a leaf component, then amount will be updated automatically. If the selected component is not a leaf component however, you can press the **Apply to all children** button (located under **Count:** entry) to apply the amount of component individuals to all leaf (grand)children of the selected component.

3.2 Setting a bounded number of instances

As mentioned in [2], in many cases it is undesirable to consider a fixed number of component's type instances, as guessing the right number may be itself a difficult task. In such cases a bounded number of instances may be used, viz. the actual number of instances should be between some lower and upper bound.

To set a bounded number of instances of a component type select the component type from the component treeview on the right-hand side. Next, on the left-hand side, under the selected component's name, tick the **Range** radiobutton and input the values of lower and upper bounds in the entries **Min:** and **Max** respectively. If the selected component is a leaf component, then bounds will be updated automatically. If the selected component is not a leaf component however, you can press the **Apply to all children** button (located under **Min:** and **Max** entries) to apply the bounded number of component individuals to all leaf (grand)children of the selected component.

3.3 Symmetry breaking

The concept of symmetry breaking can be useful to greatly reduce the number of answer sets expressing the same configuration. It ensures, that if a subset of cardinality N of a set of component's instances is admitted in a configuration, then it can be represented in **one way** only as the only possible N-element combination. For example, consider a fact:

```
component_a(1..10)
```

expressing that the configuration's instance admits 10 individuals of type *component_a*. Now assume that one of correct configurations admits 2 components of type *component_a*. That would mean that all the following subsets:

$$\{1,2\},\{1,3\},\{1,4\},...,\{2,3\},...,\{9,10\}$$

are present in some answer sets. That is a lot of redundant information! In fact that make the answer sets space explode. Thus the **symmetry** breaking option ensures that the only considered subset for the problem above would be:

$$\{1, 2\}$$

3.3.1 Setting symmetry breaking for a component

To set the symmetry breaking for a component type select the component type from the component treeview on the right-hand side. Next, on the left-hand side, under the selected component's name, tick the **Symmetry breaking** checkbox. If the selected component is a leaf component, then the property will be updated automatically. If the selected component is not a leaf component however, you can press the **Apply to children** button (located on the right of the checkbox) to apply the symmetry breaking to all to all leaf (grand)children of the selected component.

3.3.2 Setting symmetry breaking for all components

To set the symmetry breaking for all component types, tick the **Symmetry breaking for all components:** checkbox (located in the top-left corner of the tab), then press the **Apply** button (located on the right of the checkbox). Note: on default, newly created component types have the property of **Symmetry breaking** set to True. Keep that in mind when creating new component types.

4 Generation

To generate the output ASP program file select the **Run** command from the top menu bar, then select **Generate**. You will be presented with the **Generate window** where you can select the output file path for the generated ASP program and set the predicates symbol for which you wish to generate the **#show** directive.

- To select predicates symbol to generate the **#show** directive for, simply tick the checkboxes of the desired predicate symbol in the **Show selected predicates:** field.
- To show **all** answer set's predicates, tick the checkbox **Show all predicates:** located under the **Show selected predicates:** field (in that case no **#show** directive will be generated and thus entire answer set will be shown).
- To set the output ASP program file path press the **Browse** button in the **Export logic program to:** field.

5 Solving

To solve the previously generated ASP program file select the **Run** command from the top menu bar, then select **Solve**. You will be presented with the **Solve window** where you can select the input ASP program file path, the output CSV file path, the instance representation and the number of generated answer sets.

- To set the **input ASP program file path**, press the **Browse...** button in the **Select logic program** field.
- To set the **number of generated** answer sets, input the number in the **Answer sets count** field. Note: to generate **all** answer sets, input 0.
- To set the **instance representation**, select between **Id**, **Textual** and **Mixed** in the **Instance representation:** radioboxes. The difference between them is presented below:
  Assume there is a fact:

```
component_a(1..10)
```

expressing that the configuration's instance admits 10 individuals of type *component_a*. The correct configuration admits that 1 *component_a* with ID = 1 is in the answer set. Then:

Id	Textual	Mixed
in(1)	in(component_a)	in(component_a_1)

are the corresponding answer set's literals for each instance representation. Identyfying component types given an answer set can be cumbersome, if the representation type is **Id**. **Textual** representation can be useful in such cases. If one wants to preserve the component's Id and stil be able to identify the component types easily, they may utilize the **Mixed** option.

- To specify whether to ignore or obey the **#show** directives, set the **Shown predicates only:** checbox accordingly.
- To set the **exported answer sets output CSV file path**, press the **Browse...** button in the **Export answer sets to:** field.

To start the answer sets export, press **OK** button. The label above the progressbar will denote the current number of exported answer sets. You can interrupt solving by pressing the **Stop** button – answer sets generated by that time will be saved.

5.1 Generation and solving

Current configuration can be solved in just one step – by selecting **Run** > **Generate & solve...**. You will be presented with a window, combining the functionality of both **Generate window** and **Solve window**.

References

[1] T. Soininen, I. Niemelä, J. Tiihonen, and R. Sulonen. "Representing Configuration Knowledge with Weight Constraint Rules". In: (2001).

[2] G. Friedrich, A. Ryabokon, Andreas A. Falkner, Alois Haselböck, G. Schenner, and H. Schreiner. "(Re)configuration using Answer Set Programming". In: (2011).