Static GRG-Schema

The GRG Team*

Russell Bent, Carleton Coffrin, Ferdinando Fioretto, Terrence W.K. Mak, Patrick Panciatici, Pascal Van Hentenryck

Abstract

This document describes the *Grid Research for Good* (GRG) schema to represent static transmission networks.

1 Basic Concepts

This section defines the basic concepts used to model grid data. We first define data types for representing numerical values, variables, and object statuses. Then we define data types used to represent electrical values.

Throughout the document we adopt the symbol j to denote the complex imaginary unit, which satisfies the equation: $j^2 = -1$. For a complex number $x \in \mathbb{C}$ we write $\bar{x}$ to denote the conjugate of x. Additionally, for a list of elements $A = \langle a_1, \ldots, a_n \rangle$, we use the notation $A[k]$ to indicate the k^{th} element of A, a_k.

For a JSON attribute, the keyword `$ref` describes a *JSON Reference*, i.e., a reference to an object whose fragment part is a URI encoded JSON Pointer. In this document, all references are local to the schema, and their values reflect the document hierarchy.

1.1 Values

Extended Number

An *extended number* is a numeric data type whose value is either a number or one of the following: *Inf*, denoting ∞, *-Inf* denoting $-\infty$, *NaN*, standing for "not a number" and representing an undefined or unrepresentable value, or *Null*, to denote a value which is missing.

extended_number definition

```
"extended_number": {
    "oneOf": [
        {"type": "number"},
        {"enum": ["Inf", "-Inf", "NaN", "Null"]}
    ]
}
```

The following example illustrates the above concept by where *x* is assigned a numeric value, and *y* to a $-\infty$.

Example: extended_number

```
"x": 10.23,
"y": "-Inf"
```

Extended Positive Number

An *extended positive number* restricts the notion of extended number to non-negative values and the special values *Inf*, *NaN*, and *Null*.

*Contacts: Ferdinando Fioretto: fioretto@umich.edu, Pascal Van Hentenryck: pvanhent@umich.edu

extended_positive_number definition

```
"extended_positive_number": {
    "oneOf": [
        {"type": "number", "minimum": 0},
        {"enum": ["Inf", "NaN", "Null"]}
    ]
}
```

Domain

A *domain* is a collection of values used to describe valid assignments for a variable. We define three types of domains:

- Finite domains describing a collection of strings.
- Finite domains describing a collection of numbers.
- Bound domains describing a range $[lb, ub] \subseteq \mathbb{R}$.

GRG schema: domain

```
"domain": {
    "type"    : "object",
    "required": ["var"],
    "properties": {
        "var": {
            "oneOf": [
            {
                "type"    : "array",
                "items"   : {"type": "string"},
                "minItems": 1
            }, {
                "type"    : "array",
                "items"   : {"type": "number"},
                "minItems": 1
            }, {
                "type"    : "object",
                "required": ["lb", "ub"],
                "properties": {
                    "lb"  : {"$ref": "#/values/extended_number"},
                    "ub"  : {"$ref": "#/values/extended_number"}
                },
                "additionalProperties": false
            }]
        }
    },
    "additionalProperties": false
}
```

Following are examples of finite string domain (`x`), finite numerical domain (`y`), and bound domain (`z`).

Example: domain

```
"x": { "var": ["a", "b", "c"] },
"y": { "var": [1, 2, 3] },
"z": { "var": {"lb": 0, "ub": 10} }
```

Positive Domain

A *positive domain* restricts the notion of domain to non-negative values.

GRG schema: positive_domain

```
"positive_domain": {
    "type"    : "object",
    "required": ["var"],
    "properties": {
        "var": {
            "oneOf": [
            {
                "type"    : "array",
                "items"   : {"type": "string"},
```

```
                "minItems": 1
            }, {
                "type"    : "array",
                "items"   : {"type": "number", "minimum": 0},
                "minItems": 1
            }, {
                "type"    : "object",
                "required": ["lb", "ub"],
                "properties": {
                    "lb"  : {"$ref": "#/values/extended_positive_number"},
                    "ub"  : {"$ref": "#/values/extended_positive_number"}
                },
                "additionalProperties": false
            }]
        }
    },
    "additionalProperties": false
}
```

Abstract Value

An *abstract value* is an extended numeric data type which can describe either a numeric value or a variable.

GRG schema: abstract_value

```
"abstract_value": {
    "oneOf": [
        {"$ref": "#/values/extended_number"},
        {"$ref": "#/values/domain"}
    ]
}
```

In the following code, `x` is a variable abstract value, and `y` is a numeric abstract value.

Example: abstract_value

```
"x": { "var": {"lb": 0, "ub": 10} },
"y": 3.56
```

Abstract Positive Value

An *abstract positive value* restricts the notion of *abstract value* to non-negative numeric values and non negative variables.

GRG schema: abstract_positive_value

```
"abstract_value": {
    "oneOf": [
        {"$ref": "#/values/extended_positive_number"},
        {"$ref": "#/values/positive_domain"}
    ]
}
```

Status

A *status* is a special boolean variable whose domain elements are “on” and “off”.

GRG schema: abstract_status

```
"status": {
    "oneOf": [
        {"enum"       : ["off", "on"]},
        {
            "type"    : "object",
            "required": ["var"],

            "properties": {
                "var": {
```

```
                    "type"    : "array",
                    "items"   : {"enum": ["on", "off"]},
                    "minItems": 2, "maxItems": 2, "uniqueItems": true
                }
            },
            "additionalProperties": false
        }
    ]
}
```

In the following examples, `x` represents an unassigned status variable, and `y` represents a status element whose value is 'on'.

Example: status

```
"x": { "var": ["on", "off"] }
"y": "on",
```

GRG Pointer

A *GRG pointer* is a string used to identify an object's value in the GRG document. A GRG pointer extends a JSON pointer by adopting the following prefixes:

- `#`, which refers to the document root.
- `@`, which refers to a JSON object in the same scope as the pointer itself.
- `!`, which refers to network elements (a shortcut for `#/network/components`).

GRG schema: grg_pointer

```
"grg_pointer": {
    "type": "string",
    "pattern": "^(#|!|@)/.*"
}
```

In the following examples, the GRG pointers `p1`, `p2`, and `p3` refer to the same value (*br1*). The pointer `p4` refers to the first value of the array `var` in the object `status` of `switch_example`.

Example: grg_pointer

```
"network: {
    "components": {
        "switch_example" : {
            "type"    : "switch",
            "subtype" : "breaker",
            "id"      : "br1",
            "link_1"  : "10",
            "link_2"  : "11",
            "status"  : {"var" : ["off", "on"]},
            "p1"      : "@/id"
        }
    }
},

"p2": "#/network/components/switch_example/id",
"p3": "!/switch_example/id",
"p4": "!/switch_example/status/var/0
```

Table

A *table* is an object linking a list of GRG elements $\langle e_1, \ldots, e_k \rangle$ to a set of tuples $\{T_1, \ldots, T_n\}$, where each $T_i = \langle v_1, \ldots, v_k \rangle$ has values v_i (i = 1, …, k). The elements are referred to as table *arguments*, and the set of value tuples as table *values*. In other words, a table expresses the relation between a list of elements and the set of possible values for such elements.

GRG schema: table

```
"table": {
    "type"    : "object",
```

```
    "required": ["arguments", "values"],
    "properties": {
        "arguments": {
            "type"    : "array",
            "items"   : {"$ref": "#/values/grg_pointer"},
            "minItems": 1, "uniqueItems": true
        },
        "values": {
            "type" : "array":
            "items": {
                "type" : "array",
                "items": {
                    "oneOf": [
                        {"type": "#/values/abstract_value"},
                        {"type": "string"}
                    ]
                },
                "minItems": 1
            },
            "minItems": 1
        }
    }
}
```

The following example describes three assignments for the elements `x`, `y`, and `z`:

$$\mathtt{x} = 1,\ \mathtt{y} = 12,\ \mathtt{z} = 13$$
$$\mathtt{x} = 2,\ \mathtt{y} = 6,\ \mathtt{z} = 20$$
$$\mathtt{x} = 3,\ \mathtt{y} = 10,\ \mathtt{z} = 31$$

Example: table

```
"table_1" : {
    "arguments":  ["#/x", "#/y", "#/z"],
    "values"   : [[ 1,   12,  13 ],
                  [ 2,   6,   20 ],
                  [ 3,   10,  31 ]]
}
```

1.2 Electrical Values

This section defines the electrical values adopted by the GRG schema.

Impedance

Electrical *impedance* measures the opposition of a circuit to a current when a given voltage is applied. Impedance is represented as a complex quantity Z:

$$Z = R + jX, \tag{1}$$

where R denotes the resistance, and X the reactance.

Table 1 maps the real and imaginary components of impedance to their GRG schema counterparts.

GRG name	Symbol	Unit
`impedance`	Z	Ohm (Ω)
`resistance`	R	Ohm (Ω)
`reactance`	X	Ohm (Ω)

Table 1: Impedance: representation in the GRG `impedance` element and units.

GRG schema: impedance

```
"impedance": {
    "type"                 : "object",
    "required"             : ["resistance", "reactance"],
    "additionalProperties": false,
    "properties": {
        "resistance": {"$ref": "#/values/abstract_value"},
        "reactance" : {"$ref": "#/values/abstract_value"}
    }
}
```

Though the schema definition may seem complicated, the following example shows how simple it is to use:

Example: impedance

```
"impedance" : {
    "reactance" : 6.52,
    "resistance" : 2.39
}
```

Admittance

Electrical *admittance* is a measure of how much a circuit allows current to flow. Admittance is represented by a complex quantity Y:

$$Y = G + jB, \tag{2}$$

where G denotes conductance, and B denotes susceptance. These real and imaginary components are mapped to the GRG `admittance` object as shown in Table 2.

GRG name	Symbol	Unit
`admittance`	Y	Siemens (S)
`conductance`	G	Siemens (S)
`susceptance`	B	Siemens (S)

Table 2: Admittance: representation in the GRG `admittance` element and units.

GRG schema: admittance

```
"admittance": {
    "type"                 : "object",
    "required"             : ["conductance", "susceptance"],
    "additionalProperties": false,
    "properties": {
        "conductance"      : {"$ref": "#/values/abstract_value"},
        "susceptance"      : {"$ref": "#/values/abstract_value"}
    }
}
```

The following example shows how to define an admittance in GRG format:

Example: admittance

```
"shunt" : {
    "conductance" : 0,
    "susceptance" : 2.3e-05
}
```

Power

Electric power is the rate at which electrical energy is transferred by an electric circuit. We define the complex power S as:

$$S = P + jQ, \tag{3}$$

where P is the active (or real) power, and Q is the reactive power. Table 3 shows how these quantities are encoded in the GRG format.

GRG name	Symbol	Unit
`power`	S	
`active`	P	MegaWatt (*MW*)
`reactive`	Q	MegaVolt-Ampere Reactive (*MVAR*)

Table 3: Power: representation in the GRG format and units.

GRG schema: power

```
"power": {
    "type"                 : "object",
    "required"             : ["active", "reactive"],
    "additionalProperties": false,
    "properties": {
        "active"           : {"$ref": "#/values/abstract_value"},
        "reactive"         : {"$ref": "#/values/abstract_value"}
    }
}
```

Active and reactive power are defined as variables with bound domains in the following example:

Example: power

```
"power" : {
    "active"   : { "var" {"lb": 0, "ub": 30.0} },
    "reactive" : { "var" {"lb": -14.0, "ub": 14.0} }
}
```

Voltage

Voltage is the difference in electric potential energy between two points per unit electric charge. We define the voltage phasor V as:

$$V = v \cdot e^{j\theta}, \tag{4}$$

where v is the voltage magnitude, and θ is the voltage phase angle. Table 4 connects the phasor components and units to their GRG representations.

GRG name	Symbol	Unit
`voltage`	V	
`magnitude`	v	kiloVolt (kV)
`angle`	θ	Degrees

Table 4: Voltage: representation in the GRG format and units.

GRG schema: voltage

```
"voltage": {
    "type"                 : "object",
    "required"             : ["magnitude", "angle"],
    "additionalProperties": false,
    "properties": {
        "magnitude"        : {"$ref": "#/values/abstract_value"},
        "angle"            : {"$ref": "#/values/abstract_value"}
    }
}
```

In the following example, magnitude and phase angle are defined as variables:

Example: voltage

```
"voltage" : {
    "magnitude" : { "var" {"lb": 210, "ub": 250.0} },
    "angle"     : { "var" {"lb": "-Inf", "ub": "Inf"} }
}
```

1.3 Limits

Current Limits

Transmission lines (see section AC Line) and transformers (see section Transformers) have limited current-carrying capacity, described as ranges of current values $[I^{min},\ I^{max})$ that may be sustained for a duration d.

A current limit is represented in GRG format according to Table 5.

GRG name	Symbol	Unit
`min`	I^{min}	Ampere (A)
`max`	I^{max}	Ampere (A)
`duration`	d	Seconds (sec)

Table 5: Current Limits: representation in the GRG format and units.

A current limit object is expressed as a *table* whose *arguments* are `duration`, `min`, `max`, and `report`. The `report` argument is a boolean field indicating whether the status of the branch should be signaled to the operator ('on') or not ('off').

GRG schema: current_limits

```
"current_limits": {
    "type"     : "object",
    "required": ["position", "steps"],
    "additionalProperties": false,
    "properties": {
        "arguments": {
            "type" : "array",
            "items": [
                {"enum":["duration"]}, {"enum":["min"]}, {"enum":["max"]}, {"enum":["report"]}
             ],
             "minItems": 4, "maxItems": 4, "additionalItems": "false"
         },
         "values": {
             "type" : "array",
             "items": [
                 {
                 "type": "array",
                 "items": [
                     {"$ref": "#/values/extended_number"},
                     {"$ref": "#/values/extended_number"},
                     {"$ref": "#/values/extended_number"},
                     {"enum": ["on", "off"]}
                 ],
                 "minItems": 4, "maxItems": 4, "additionalItems": "false"
             }
          ],
          "minItemes": 1
       }
    }
}
```

An example of current limits is provided below. The branch can (1) carry up to 563 A indefinitely, or (2) carry between 563 A and 746 A indefinitely, but with a signal to the system operators, or (3) carry current in excess of 746 A for at most 6300 seconds before tripping the branch.

Example: current_limits

```
"current_limits" : {
    "arguments":  ["duration", "min", "max", "report"},
    "values"    : [["Inf",   0,  563,  "off"],
                   ["Inf", 563,  746,  "on"],
                   [6300,  746, "Inf", "off"]]
}
```

Thermal Limits

Transmission line and transformer limits can also be described in terms of power $[P^{min}, P^{max})$ that may be safely carried for a given duration d. The GRG representation of a thermal limit is provided in Table 6.

GRG name	Symbol	Unit
`min`	S^{min}	MegaWatt (*MW*)
`max`	S^{max}	MegaWatt (*MW*)
`duration`	d	Seconds (*sec*)

Table 6: Thermal Limits: representation in the GRG format and units.

Similar to the current limit, a thermal limit object is expressed as a *table* whose *arguments* are: `duration`, `min`, `max`, and `report`. As with the current limit, `report` is a boolean indicating whether the status of the branch should be signaled to the operator ('on') or not ('off').

GRG schema: thermal_limits

```
"thermal_limits": {
    "type"    : "object",
    "required": ["position", "steps"],
    "additionalProperties": false,
    "properties": {
        "arguments": {
            "type" : "array",
            "items": [
                {"enum":["duration"]}, {"enum":["min"]}, {"enum":["max"]}, {"enum":["report"]}
             ],
             "minItems": 4, "maxItems": 4, "additionalItems": "false"
         },
         "values": {
             "type" : "array",
             "items": [
                 {
                 "type": "array",
                 "items": [
                     {"$ref": "#/values/extended_number"},
                     {"$ref": "#/values/extended_number"},
                     {"$ref": "#/values/extended_number"},
                     {"enum": ["on", "off"]}
                 ],
                 "minItems": 4, "maxItems": 4, "additionalItems": "false"
             }
          ],
          "minItemes": 1
       }
    }
}
```

2 Network Components

Having described the fundamental parameters of electrical devices like transmission lines, transformers, and generators, we now describe these components themselves.

AC Line

An AC line connects network devices in two different points of the network. It has two sides: *side 1* is defined as the sending side, and *side 2* is defined as the receiving side. The current is positive when flowing from side 1 to side 2.

The currents I_1 (at side 1) and I_2 (at side 2) are given by:

$$I_1 = Y_1 \cdot V_1 + \frac{1}{Z}(V_1 - V_2) \tag{5}$$

$$I_2 = Y_2 \cdot V_2 + \frac{1}{Z}(V_1 - V_2), \tag{6}$$

where $Y_1 = G_1 + jB_1$ and $Y_2 = G_2 + jB_2$ are the admittances at sides 1 and 2, respectively, $Z = R + jX$ is the line impedance, and V_1 and V_2 are the voltages at the connecting points. Figure 1 illustrates an AC line between nodes 1 and 2 (represented by the black points at the ends).

The power S_i at side i ($i \in \{1, 2\}$) is given by:

$$S_i = \bar{I}_i \cdot V_i \tag{7}$$

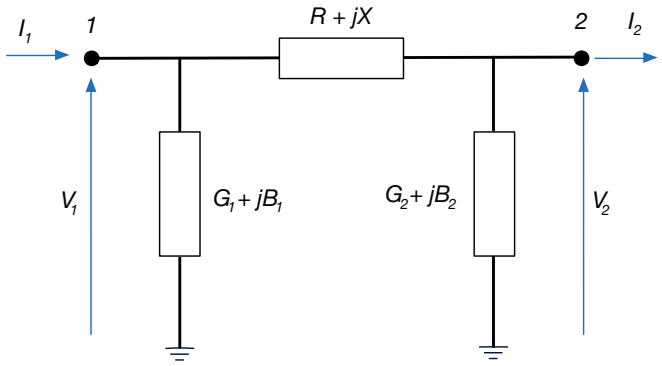

Figure 1: Illustration of an AC line.

Line current limits are monitored at both ends. The absolute value of current magnitude is compared to a sequence of current limits $L_1 = L_{1_1}, \ldots, L_{1_n}$, and $L_2 = L_{2_1}, \ldots, L_{2_n}$, where each L_{i_k} ($i \in \{1, 2\}, k \in \{1, \ldots, n\}$) is a current limit object, and for each L_{i_k}, ($k > 1$), $I^{min}_{i_k} = I^{max}_{i_{k-1}}$. Finally, the current limit durations are such that $d_{i_1} = \infty$, denoting that L_{i_1} is a permanent acceptable current limit.

GRG schema: ac_line

```
"ac_line": {
    "type"     : "object",
    "required": ["type", "id", "link_1", "link_2", "voltage_level_1_id", "voltage_level_2_id",
                 "shunt_1", "shunt_2", "impedance", "current_limits_1", "current_limits_2"],

    "properties": {
        "type"               : {"enum": ["ac_line"]},
        "id"                 : {"type": "string"},
        "description"        : {"type": "string"},
        "link_1"             : {"type": "string"},
        "link_2"             : {"type": "string"},
        "voltage_level_1_id": {"type": "string"},
        "voltage_level_2_id": {"type": "string"},
        "shunt_1"            : {"$ref": "#/electrical_values/admittance"},
        "shunt_2"            : {"$ref": "#/electrical_values/admittance"},
        "impedance"          : {"$ref": "#/electrical_values/impedance"},
        "current_limits_1"   : {"$ref": "#/limits/current_limits"},
        "current_limits_2"   : {"$ref": "#/limits/current_limits"},
        "thermal_limits_1"   : {"$ref": "#/limits/thermal_limits"},
        "thermal_limits_2"   : {"$ref": "#/limits/thermal_limits"}
    }
}
```

In the AC line GRG schema, `type` identifies the type of the object (i.e., an *AC line*), `id` is an unique identifier for the network component, and `description` is an optional field which can be used to describe the component. The

fields `link_1` and `link_2` describe the identifiers of the connecting network elements at the side 1 and side 2, respectively, of the line. `voltage_level_1_id` and `voltage_level_2_id` describe the identifiers of the voltage levels connected by the line. `shunt_1` and `shunt_2` define the admittances Y_1 and Y_2 at sides 1 and 2,respectively. `impedance` defines the impedance Z of the line. `current_limits_1` and `current_limits_2` describe collections of current limits associated to sides 1 and 2 of the line. Finally, `thermal_limits_1` and `thermal_limits_2` describe collections of thermal limits associated to sides 1 and 2 of the line. These last components are optional. A summary of these AC line components is provided in Table 7.

GRG name	symbol	unit
`link_1`	side 1	
`link_2`	side 2	
`impedance`→`resistance`	R	Ohm (Ω)
`impedance`→`reactance`	X	Ohm (Ω)
`shunt_1`→`conductance`	G_1	Siemens (S)
`shunt_1`→`susceptance`	B_1	Siemens (S)
`shunt_2`→`conductance`	G_2	Siemens (S)
`shunt_2`→`susceptance`	B_2	Siemens (S)
`current_limits_1`→`values[k][0]`	d_{1_k}	Seconds (*sec*)
`current_limits_1`→`values[k][1]`	$I_{1_k}^{min}$	Ampere (A)
`current_limits_1`→`values[k][2]`	$I_{1_k}^{max}$	Ampere (A)
`current_limits_2`→`values[k][0]`	d_{2_k}	Seconds (*sec*)
`current_limits_2`→`values[k][1]`	$I_{2_k}^{min}$	Ampere (A)
`current_limits_2`→`values[k][2]`	$I_{2_k}^{max}$	Ampere (A)
`thermal_limits_1`→`values[k][0]`	d_{1_k}	Seconds (*sec*)
`thermal_limits_1`→`values[k][1]`	$S_{1_k}^{min}$	MegaWatt (MW)
`thermal_limits_1`→`values[k][2]`	$S_{1_k}^{max}$	MegaWatt (MW)
`thermal_limits_2`→`values[k][0]`	d_{2_k}	Seconds (*sec*)
`thermal_limits_2`→`values[k][1]`	$S_{2_k}^{min}$	MegaWatt (MW)
`thermal_limits_2`→`values[k][2]`	$S_{2_k}^{max}$	MegaWatt (MW)

Table 7: AC line: representation in the GRG format and units.

An example of an AC line in GRG format is provided in Figure 2.

Example: AC Line

```
"line_ln1" : {
    "type"              : "line",
    "id"                : "ln1",
    "link_1"            : "9",
    "link_2"            : "3",
    "voltage_level_1_id": "vl1",
    "voltage_level_2_id": "vl2",
    "current_limits_1" : {
        "arguments":  ["duration", "min", "max", "report"},
        "values"   : [["Inf",   0,  563,  "off"],
                      ["Inf", 563,  746,  "on"],
                      [6300,  746, "Inf", "off"]]
    },
    "current_limits_2" : {
        "arguments":  ["duration", "min", "max", "report"},
        "values"   : [["Inf",   0,  563,  "off"]]
    },
    "impedance"    : {"reactance" : 6.52, "resistance" : 2.39},
    "shunt_1"      : {"conductance" : 0, "susceptance" : 2.3e-05},
    "shunt_2"      : {"conductance" : 0, "susceptance" : 2.3e-05}
}
```

Figure 2: An example of a GRG AC Line.

2.1 Transformers

Two Winding Transformer

A two winding transformer connects devices located at two different voltage levels of the network. We denote these voltage levels VL_1 and VL_2, and their associated nominal voltage values v_1^{nom} and v_2^{nom}.

Most transformers come equipped with taps on their winding to adjust either the voltage transformation or the reactive flow through the transformer. The transformer voltage magnitude ratio r_k, phase shift δ_k, impedance Z_k, and admittance Y_k all depend on the tap k.

We define *side 1* as the high voltage side, and *side 2* as the low voltage side. The current is positive when flowing from 1 to 2. The currents at sides 1 and 2 are:

$$I_1 = \rho'_k \cdot \left(Y_k \cdot V'_1 + \frac{1}{Z_k}(V'_1 - V_2)\right) \tag{8}$$

$$I_2 = \frac{1}{Z_k}(V_2 - V'_1), \tag{9}$$

where $Y_k = G_k + jB_k$ is the admittance at side 1 of the transformer on tap k, $Z_k = R_k + jX_k$ is the impedance of the transformer on tap k, V_1 and V_2 are the voltages at the connecting buses in VL_1 and VL_2, respectively. V'_1 is defined as $(\rho'_k \cdot V_1)$, and ρ'_k is a complex ratio defined for the k-th tap as:

$$\rho'_k = \left(\left(\frac{V_2^{\text{nom}}}{V_1^{\text{nom}}}\right) r_k\right) \cdot e^{j\delta_k}, \tag{10}$$

where V_1^{nom} and V_2^{nom} denote nominal voltage magnitudes at sides 1 and 2 of the transformer, respectively. Figure 3 illustrates a two winding transformer, where the points labeled 1 and 2 represent sides 1 and 2 connected by the branch.

The power S_i at sides i ($i \in \{1, 2\}$) is given by:

$$S_i = \bar{I}_i \cdot V_i \tag{11}$$

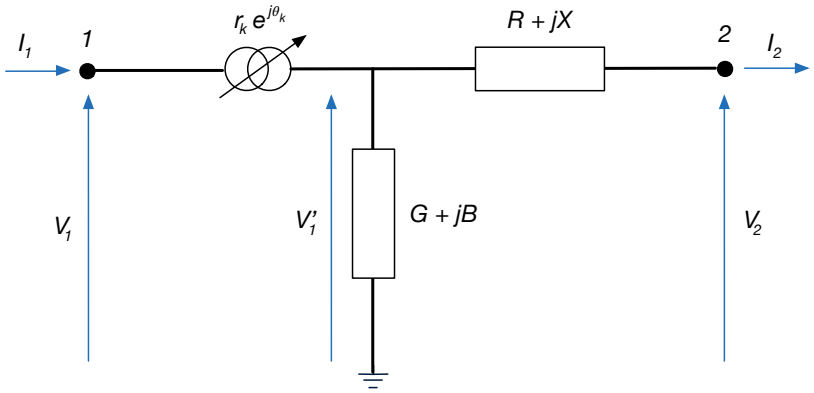

Figure 3: Illustration of a two winding transformer.

As with AC lines, transformers have current capacity limits, and the maximum absolute value of the current magnitude is monitored on both sides of the branch. Constraints are described through a sequence of current limits $L_1 = L_{1_1}, \ldots, L_{1_n}$, and $L_2 = L_{2_1}, \ldots, L_{2_n}$, where each L_{i_k} ($i \in \{1, 2\}, k \in \{1, \ldots, n\}$) is a current limit object, and for each L_{i_k}, $(k > 1)$, $I_{i_k}^{min} = I_{i_{k-1}}^{max}$. Finally, the current limit durations are such that $d_{i_1} = \infty$, denoting that L_{i_1} is a permanent acceptable current limit.

GRG schema: two_winding_transformer

```
"two_winding_transformer": {
    "type": "object",
    "required": ["type", "id", "link_1", "link_2", "voltage_level_1_id", "voltage_level_2_id",
                 "tap_changer"],
```

```
    "additionalProperties": false,
    "properties": {
        "type"               : {"enum": ["two_winding_transformer"]},
        "id"                 : {"type": "string"},
        "description"        : {"type": "string"},
        "link_1"             : {"type": "string"},
        "link_2"             : {"type": "string"},
        "voltage_level_1_id" : {"type": "string"},
        "voltage_level_2_id" : {"type": "string"},
        "current_limits_1"   : {"$ref": "#/limits/current_limits"},
        "current_limits_2"   : {"$ref": "#/limits/current_limits"},
        "thermal_limits_1"   : {"$ref": "#/limits/thermal_limits"},
        "thermal_limits_2"   : {"$ref": "#/limits/thermal_limits"},

        "tap_changer": {
            "type": "object",
            "required"       : ["position", "impedance", "shunt", "tap_ratio", "angle_shift",
                                "steps"],
            "additionalProperties": false,
            "properties": {
                "position"   : {"$ref": "#/values/abstract_value"},
                "impedance"  : {"$ref": "#/electrical_values/impedance"},
                "shunt"      : {"$ref": "#/electrical_values/admittance"},
                "tap_ratio"  : {"$ref": "#/values/abstract_value"},
                "angle_shift": {"$ref": "#/values/abstract_value"},
                "steps"      : {"$ref": "#/values/table"}
            }
        }
    }
}
```

In the two winding transformer GRG schema, `type` identifies the type of the object (i.e., a *two winding transformer*), `id` is an unique identifier for the network component, and `description` is an optional field which can be used to describe the component. The fields `link_1` and `link_2` identify the connecting network elements at sides 1 and side 2 of the transformer; `voltage_level_1_id` and `voltage_level_2_id` describe the identifiers of the voltage levels connected by the branch. `current_limits_1` and `current_limits_2` describe collections of current limits associated to sides 1 and 2 of the line. The optional `thermal_limits_1` and `thermal_limits_2` describe collections of thermal limits associated to sides 1 and 2 of the line. Finally, `tap_changer` describes the transformer taps. The `position` field describes the index of the tap step on which the transformer is positioned, and the `steps` is a table object describing the possible tuple assignments for the position index, the impedance (`impedance`), the admittance (`shunt`), the tap magnitude (`tap_ratio`) and phase shift (`angle_shift`) of the transformer's tap.

A summary of the two winding transformer components is provided in Table 8, and an example is provided in Figure 4.

Three winding Transformer

A three winding transformer connects devices located at three different voltage levels of the network. We denote these voltage levels VL_1, VL_2, and VL_3 and their associated nominal voltage values as v_1^{nom}, v_2^{nom}, and v_3^{nom}, respectively.

We define *Side 1* as the high voltage side, *side 2* as the medium voltage side, and *side 3* as the low voltage side. The current is positive when flowing from 1 to 2 or 3.

The current at sides 1, 2, and 3 is given, respectively by:

$$I_1 = Y_k \cdot V_m + \frac{1}{Z_1}(V_1 - V_m) \tag{12}$$

$$I_2 = \frac{1}{\rho_{2_k}} \cdot \frac{1}{Z_{2_k}}(V_2 - V_m) \tag{13}$$

$$I_3 = \frac{1}{\rho_{3_k}} \cdot \frac{1}{Z_{3_k}}(V_3 - V_m), \tag{14}$$

GRG name	symbol	unit
link_1	side 1	
link_2	side 2	
current_limits_1→values[k][0]	d_{1_k}	Seconds (*sec*)
current_limits_1→values[k][1]	$I_{1_k}^{min}$	Ampere (*A*)
current_limits_1→values[k][2]	$I_{1_k}^{max}$	Ampere (*A*)
current_limits_2→values[k][0]	d_{2_k}	Seconds (*sec*)
current_limits_2→values[k][1]	$I_{2_k}^{min}$	Ampere (*A*)
current_limits_2→values[k][2]	$I_{2_k}^{max}$	Ampere (*A*)
thermal_limits_1→values[k][0]	d_{1_k}	Seconds (*sec*)
thermal_limits_1→values[k][1]	$S_{1_k}^{min}$	MegaWatt (*MW*)
thermal_limits_1→values[k][2]	$S_{1_k}^{max}$	MegaWatt (*MW*)
thermal_limits_2→values[k][0]	d_{2_k}	Seconds (*sec*)
thermal_limits_2→values[k][1]	$S_{2_k}^{min}$	MegaWatt (*MW*)
thermal_limits_2→values[k][2]	$S_{2_k}^{max}$	MegaWatt (*MW*)
tap_changer→position	k	
tap_changer→impedance→resistance	R_k	Ohm (Ω)
tap_changer→impedance→reactance	X_k	Ohm (Ω)
tap_changer→shunt→conductance	G_k	Siemens (S)
tap_changer→shunt→susceptance	B_k	Siemens (S)
tap_changer→tap_ratio	r_k	per unit
tap_changer→angle_shift	δ_k	degrees

Table 8: Two winding transformer: representation in the GRG format and units.

where $Y_k = G_k + jB_k$ is the admittance at side 1 of the transformer with tap k; $Z_{2_k} = R_2 + jX_2$ and $Z_{3_k} = R_3 + jX_3$ are the impedance values of sides 2 and 3 of the transformer with tap k; V_1, V_2, and V_3 are the voltages at the connecting buses; V_m is the voltage phasor in the star middle point; ρ_{2_k} and ρ_{3_k} are the complex ratios on sides 2 and 3 of the transformer, defined as:

$$\rho'_{2_k} = \left((\frac{V_2^{\text{nom}}}{V_1^{\text{nom}}})r_{2_k}\right) \cdot e^{j\delta_{2_k}} \tag{15}$$

$$\rho'_{3_k} = \left((\frac{V_3^{\text{nom}}}{V_1^{\text{nom}}})r_{3_k}\right) \cdot e^{j\delta_{3_k}}, \tag{16}$$

corresponding to the k-th tap; and V_1^{nom}, V_2^{nom}, and V_3^{nom} denote the nominal voltage magnitudes at sides 1, 2, and 3 of the transformer. Figure 5 provides an illustration of a three winding transformer.

The power S_i at sides i ($i \in \{1, 2, 3\}$) is given by:

$$S_i = \bar{I}_i \cdot V_i \tag{17}$$

GRG schema: three_winding_transformer

```
"three_winding_transformer": {
    "type": "object",
    "required": ["type", "id", "link_1", "link_2", "link_3", "voltage_level_1_id",
                 "voltage_level_2_id", "voltage_level_3_id", "impedance_1", "shunt"],
    "additionalProperties": false,
    "properties": {
        "type"              : {"enum": ["three_winding_transformer"]},
        "id"                : {"type": "string"},
        "description"       : {"type": "string"},
        "link_1"            : {"type": "string"},
        "link_2"            : {"type": "string"},
        "link_3"            : {"type": "string"},
        "voltage_level_1_id": {"type": "string"},
        "voltage_level_2_id": {"type": "string"},
        "voltage_level_3_id": {"type": "string"},
```

Example: Two Winding Transformer

```
"two_winding_transformer_tr1" : {
    "type" : "two_winding_transformer",
    "id"                : "tr1",
    "link_1"            : "11",
    "link_2"            : "3",
    "voltage_level_1_id": "vl2",
    "voltage_level_2_id": "vl3",
    "current_limits_1" : {
        "arguments": ["duration", "min", "max", "report"],
        "values"   :   [
            ["Inf", 0,    1029,  "off"],
            [1200,  1029, 1342,  "off"],
            [300,   1342, 1790,  "off"],
            [60,    1790, "Inf", "off"]
        ]
     },
    "tap_changer"    : {
        "position"   : {"var" : [0, 1, 2]},
        "impedance"  : {"reactance": {"var": [ 60.1 ] }, "resistance": {"var": [ 1.512] }},
        "shunt"      : {"conductance": {"var": [0] },  "susceptance": {"var": [0] }},
        "tap_ratio"  : {"var": [0.988, 1.00, 1.011]},
        "angle_shift": {"var": [0]},
        "steps": {
            "arguments": ["@/tap_changer/position", "@/tap_changer/impedance",
                          "@/tap_changer/shunt", "@/tap_changer/tap_ratio", "@/tap_changer/angle_shift"],
            "values"   : [
            [0, {"reactance":60.1, "resistance":1.512}, {"conductance":0, "susceptance":0}, 0.988, 0],
            [1, {"reactance":60.1, "resistance":1.512}, {"conductance":0, "susceptance":0}, 1.0,   0],
            [2, {"reactance":60.1, "resistance":1.512}, {"conductance":0, "susceptance":0}, 1.011, 0]
            ]
        }
    }
}
```

Figure 4: An example of a GRG Two Winding Transformer.

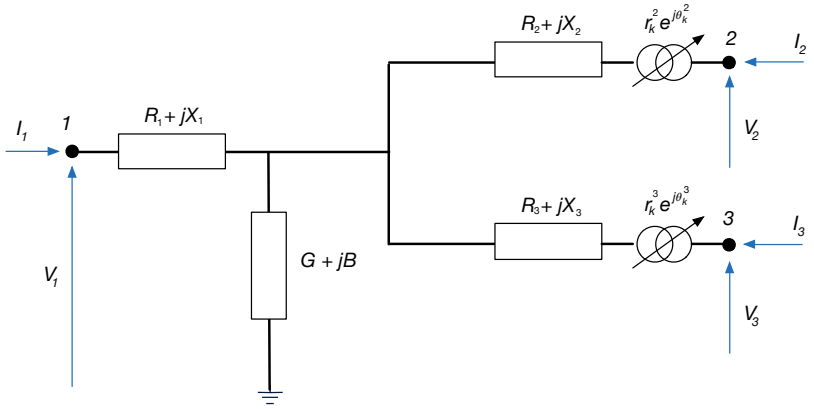

Figure 5: Illustration of a three winding transformer.

```
    "current_limits_1"  : {"$ref": "#/limits/current_limits"},
    "current_limits_2"  : {"$ref": "#/limits/current_limits"},
    "current_limits_3"  : {"$ref": "#/limits/current_limits"},
    "thermal_limits_1"   : {"$ref": "#/limits/thermal_limits"},
    "thermal_limits_2"   : {"$ref": "#/limits/thermal_limits"},
    "thermal_limits_3"   : {"$ref": "#/limits/thermal_limits"},

    "shunt"             : {"$ref": "#/electrical_values/admittance_rectangular"},
    "impedance_1"       : {"$ref": "#/electrical_values/impedance_rectangular"},
```

```
        "tap_changer": {
            "type": "object",
            "required": ["position", "steps"],
            "additionalProperties": false,
            "properties": {
                "position"     : {"$ref": "#/values/abstract_value"},
                "impedance_2"  : {"$ref": "#/electrical_values/impedance"},
                "impedance_3"  : {"$ref": "#/electrical_values/impedance"},
                "shunt_2"      : {"$ref": "#/electrical_values/admittance"},
                "shunt_3"      : {"$ref": "#/electrical_values/admittance"},
                "tap_ratio_2"  : {"$ref": "#/values/abstract_value"},
                "tap_ratio_3"  : {"$ref": "#/values/abstract_value"},
                "angle_shift_2": {"$ref": "#/values/abstract_value"},
                "angle_shift_3": {"$ref": "#/values/abstract_value"},
                "steps"        : {"$ref" : "#/values/table"}
            }
        }
    }
}
```

In the three winding transformer GRG schema, `type` identifies the type of the object (i.e., a *three windings transformer*), `id` is an unique identifier for the network component, and `description` is an optional field which can be used to describe the component. The fields `link_1`, `link_2`, and `link_3` identify connecting network elements at sides 1, 2 and 3 of the transformer; `voltage_level_1_id`, `voltage_level_2_id`, and `voltage_level_3_id` identify voltage levels connected to the branch; `shunt` defines the admittance Y; `impedance_1` defines the impedance Z_1 of branch 1; and `tap_changer` describes the transformer taps. `position` represents the index of the current tap step, and `steps` is a table with tuple assignments for position index, `impedance_2` and `impedance_3`, `tap_ratio_2` and `tap_ratio_3`, and phase shifts `angle_shift_2` and `angle_shift_3` corresponding to the k^{th} tap.

A summary of three windings transformer components is provided in Table 9, and Figure 6 provides an example.

Switch

A switch is an electrical component that can interrupt the current in a circuit. There are two types of switches: *circuit breakers* and *disconnectors*. Circuit breakers can be switched on or off when they are energized, while disconnectors can be switched only when not energized. These are series devices with two sides denoted side 1 and side 2. Their operation satisfies the following:

$$s \cdot V_1 = s \cdot V_2, \tag{18}$$

where s is a binary variable denoting the switch status (1 for open, 0 for closed), and V_1 and V_2 are the voltages at sides 1 and 2 of the switch.

Figure 7 illustrates an open disconnector (a) and breaker (c), and a closed disconnector (b) and breaker (d).

GRG schema: switch

```
"switch": {
    "type"    : "object",
    "required": ["type", "subtype", "id", "status", "link_1", "link_2"],
    "additionalProperties": false,
    "properties": {
        "type"       : {"enum": ["switch"]},
        "subtype"    : {"enum": ["breaker", "disconnector"]},
        "description": {"type": "string"},
        "id"         : {"type": "string"},
        "status"     : {"$ref": "#/values/abstract_status"},
        "link_1"     : {"type": "string"},
        "link_2"     : {"type": "string"}
    }
}
```

The field `type` identifies the type of the object (`switch`), `subtype` denotes whether the switch is a breaker or a disconnector, `description` is an optional field for describing the switch, `id` is a unique identifier, `status` denotes

GRG name	symbol	unit
link_1	side 1	
link_2	side 2	
impedance_1→resistance	R_1	Ohm (Ω)
impedance_1→reactance	X_1	Ohm (Ω)
shunt→conductance	G	Siemens (S)
shunt→susceptance	B	Siemens (S)
current_limits_1→values[k][0]	d_{1_k}	Seconds (*sec*)
current_limits_1→values[k][1]	$I_{1_k}^{min}$	Ampere (*A*)
current_limits_1→values[k][2]	$I_{1_k}^{max}$	Ampere (*A*)
current_limits_2→values[k][0]	d_{2_k}	Seconds (*sec*)
current_limits_2→values[k][1]	$I_{2_k}^{min}$	Ampere (*A*)
current_limits_2→values[k][2]	$I_{2_k}^{max}$	Ampere (*A*)
current_limits_3→values[k][0]	d_{3_k}	Seconds (*sec*)
current_limits_3→values[k][1]	$I_{3_k}^{min}$	Ampere (*A*)
current_limits_3→values[k][2]	$I_{3_k}^{max}$	Ampere (*A*)
thermal_limits_1→values[k][0]	d_{1_k}	Seconds (*sec*)
thermal_limits_1→values[k][1]	$S_{1_k}^{min}$	MegaWatt (*MW*)
thermal_limits_1→values[k][2]	$S_{1_k}^{max}$	MegaWatt (*MW*)
thermal_limits_2→values[k][0]	d_{2_k}	Seconds (*sec*)
thermal_limits_2→values[k][1]	$S_{2_k}^{min}$	MegaWatt (*MW*)
thermal_limits_2→values[k][2]	$S_{2_k}^{max}$	MegaWatt (*MW*)
thermal_limits_3→values[k][0]	d_{3_k}	Seconds (*sec*)
thermal_limits_3→values[k][1]	$S_{3_k}^{min}$	MegaWatt (*MW*)
thermal_limits_3→values[k][2]	$S_{3_k}^{max}$	MegaWatt (*MW*)
tap_changer→impedance_3→resistance	R_k^3	Ohm (Ω)
tap_changer→impedance_3→reactance	X_k^3	Ohm (Ω)
tap_changer→tap_ratio_3	r_k^3	per unit
tap_changer→angle_shift_3	δ_k^3	degrees

Table 9: Two windings transformer: representation in the GRG format and units.

whether the switch is open or closed, and link_1 and link_2 identify the connecting network elements at sides 1 and side 2 of the switch.

Table 10 summarizes switch components, and Figure 8 provides an example.

GRG name	symbol	unit
status	s	boolean

Table 10: Switch representation in GRG format and units.

Bus

A bus is a set of equipment connected together. It could be a configured object or the result of a computation, depending of the context.

GRG schema: bus

```
"bus": {
    "type"    : "object",
    "required": ["type", "id", "voltage"],
    "additionalProperties": false,
    "properties": {
        "type"       : {"enum": ["busbar"|"logical_bus"|"bus"]},
        "description": {"type": "string"},
        "id"         : {"type": "string"},
```

Example: Three Windings Transformer

```
"three_windings_transformer_tr1" : {
    "type"              : "three_windings_transformer",
    "id"                : "tr1",
    "link_1"            : "2",
    "link_2"            : "3",
    "link_3"            : "6",
    "voltage_level_1_id": "vl1",
    "voltage_level_2_id": "vl2",
    "voltage_level_1_id": "vl3",
    "impedance_1"       : {"reactance" : 60.1, "resistance" : 1.512},
    "shunt"             : {"conductance" : 0, "susceptance" : 0},
    "tap_changer" : {
        "position"      : {"var" : [0]},
        "impedance_2"   : {"reactance": {"var": [ 60.1 ] }, "resistance": {"var": [ 1.512] }},
        "impedance_3"   : {"reactance": {"var": [ 60.1 ] }, "resistance": {"var": [ 1.512] }},
        "tap_ratio_2"   : {"var": [1.00]},
        "tap_ratio_3"   : {"var": [1.00]},
        "angle_shift_2" : {"var": [0]},
        "angle_shift_3" : {"var": [0]},

        "steps": {
            "arguments" : ["@tap_changer/position",
                           "@tap_changer/impedance_2",  "@tap_changer/impedance_3",
                           "@tap_changer/tap_ratio_2", "@tap_changer/tap_ratio_3",
                           "@tap_changer/angle_shift_2", "@tap_changer/angle_shift_3"],
            "values"    : [
            [0, {"reactance":60.1, "resistance":1.512}, {"reactance":60.1, "resistance":1.512},
                1.00, 1.00, 0, 0]
            ]
        }
    }
}
```

Figure 6: An example of a GRG Three Windings Transformer.

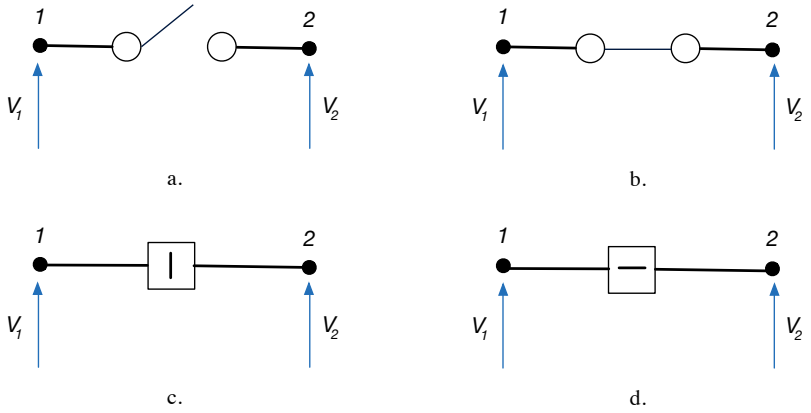

Figure 7: Illustration of an open (a) and close (b) disconnector, and an open (c) and close (d) breaker.

```
        "link"       : {"type": "string"}
        "voltage"    : {"type" : "#/electrical_values/voltage"}
        "name"       : {"type": "string"},
    }
}
```

`type` identifies the type of the object. Depending on the network topology adopted, a bus can be represented as a `busbar` (node-breaker topology), `logical bus` (bus-breaker topology), or simply `bus` (bus-branch topology). `description` is an optional description field, and `id` is an unique identifier. The field `link` is active exclusively

Example: Switch

```
"switch_vl1_br1" : {
    "type"   : "switch",
    "subtype": "breaker",
    "id"     : "br1",
    "link_1" : "10",
    "link_2" : "11",
    "status" : {"var" : ["off", "on"]}
}
```

Figure 8: An example of a GRG switch.

in the node-breaker and bus-breaker network topologies; it identifies the node at which the bus is connected. Finally, `voltage` refers to the bus voltage magnitude v and phase angle θ, and and `name` denotes the bus name.

Table 11 tabulates the bus components, while Figure 9 contains an example of a bus in GRG format.

GRG name	symbol	unit
`voltage`→`magnitude`	v	kiloVolt (kV)
`voltage`→`angle`→`lb`	Θ^l	Degrees
`voltage`→`angle`→`ub`	Θ^u	Degrees

Table 11: Bus representation in GRG format and units.

Example: Busbar

```
"busbar_vl1_bbr1" : {
    "type" : "busbar",
    "id"   : "bbr1",
    "name" : "E",
    "link" : "0",
    "voltage" : {
        "angle"     : {"var" : {"lb" : -30.0, "ub" : 30.0}},
        "magnitude" : {"var" : {"lb" : "-Inf", "ub" : "Inf"}}
    }
}
```

Figure 9: An example of a GRG busbar.

Shunt

A shunt capacitor or reactor is defined as an admittance:

$$I = -Y \cdot V,$$

where $Y = G + jB$ is the admittance, and V is the voltages at the connecting point. Figure 10 illustrates a shunt.

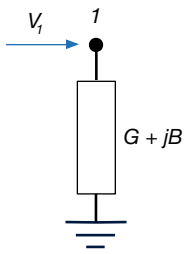

Figure 10: A shunt.

GRG schema: shunt

```
"shunt": {
    "type"     : "object",
    "required": ["type", "id", "link", "shunt"],
    "additionalProperties": false,
    "properties": {
        "type"        : {"enum": ["shunt"]},
        "id"          : {"type": "string"},
        "description": {"type": "string"},
        "link"        : {"type": "string"},
        "shunt"       : {"$ref": "#/electrical_values/admittance"}
    }
}
```

In the above schema, `type` identifies the type of the object (i.e., a `shunt`), `id` is a unique identifier, and `description` is an optional description field. The field `link` identifies the connecting network element (it can be either a node, if the network is in node-breaker topology or bus-breaker topology, or a bus, if the network is in a bus-branch topology). Finally, `shunt` defines the admittance Y.

Table 12 summarizes shunt components, while Figure 11 provides an example.

GRG name	symbol	unit
shunt→conductance	G	Siemens (S)
shunt→susceptance	B	Siemens (S)

Table 12: Shunt representation in the GRG format and units.

Example: Shunt

```
"shunt_vl1_s1" : {
    "id"   : s1
    "type" : "shunt",
    "link" : "7",
    "shunt": {"conductance" : 0, "susceptance" : 0.005039}
}
```

Figure 11: An example of a GRG shunt.

Load

A Load consumes active power P and reactive power Q at its connection point, as illustrated in Figure 12.

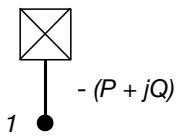

Figure 12: A load.

load definition

```
"load": {
    "type": "object",
    "required": ["type", "id", "link", "demand"],
    "additionalProperties": false,
    "properties": {
        "type"        : {"enum": ["load"]},
        "subtype"     : {"enum": ["auxiliary", "undefined"]},
        "id"          : {"type": "string"},
        "description": {"type": "string"},
```

```
        "link"       : {"type": "string"},
        "demand"     : {"$ref": "#/electrical_values/power"},
    }
}
```

In the load GRG schema, `type` identifies the type of the object (i.e., a `load`), `subtype` identifies the type of load: (`auxiliary` or `undefined`). The attribute `id` is an unique identifier, and `description` is an optional description field. The field `link` identifies the connecting network element (it can be either a node, if the network is in node-breaker topology or bus-breaker topology, or a bus, if the network is in a bus-branch topology). Finally, `demand` defines the power S consumed by the load.

Table 13 summarizes load components, while Figure 13 provides an example.

GRG name	symbol	unit
demand→active	P	MegaWatt (*MW*)
demand→reactive	Q	MegaVolt-Ampere Reactive (*MVAR*)

Table 13: Load representation in GRG format and units.

Example: Load

```
"load_vl1_ld1" : {
    "type"   : "load"
    "subtype": "undefined",
    "id"     : "ld1",
    "link"   : "7",
    "demand" : {
        "active"   : {"var" : {"lb" : "-Inf", "ub" : "Inf"}},
        "reactive" : {"var" : {"lb" : "-Inf", "ub" : "Inf"}}
    }
}
```

Figure 13: An example of a GRG load.

Generator

A generator produces active power (P) and reactive power (Q), and supports voltage (V). Its output is limited according to a PQ-curve, which specifies minimum and maximum reactive power values for every active power value. More formally, a PQ-curve is a sequence of tuples: $\langle P_k, Q_k^{min}, Q_k^{max} \rangle_{k=1}^{n}$ (with $n > 0$) such that for each $k < n$, $P_k < P_{k+1}$, and $Q_k^{min} \leq Q_k^{max}$.

Figure 14 illustrates a generator (a) and feasible PQ region (shaded gray area) of its PQ-curve.

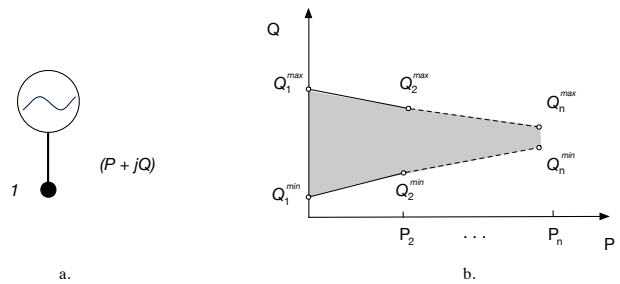

Figure 14: Illustration of a generator (a) and its PQ-curve (b).

GRG schema: generator

```
"generator": {
    "type"    : "object",
```

```
    "required": ["type", "id", "link", "output"],
    "additionalProperties": false,
    "properties": {
        "type"   : {"enum": ["generator"]},
        "subtype": {"enum": ["hydro", "wind", "thermal", "other", "nuclear", "solar"]},
        "id"     : {"type": "string"},
        "link"   : {"type": "string"},
        "output" : {"$ref": "#/electrical_values/power"},
        "PQ_curve"   : {
            "type": "object",
            "required": ["arguments", "values"],
            "additionalProperties": false,
            "properties": {
                "arguments": {
                    "type": "array",
                    "items": [
                        {"enum": ["active"]}, {"enum": ["reactive_lb"]}, {"enum": ["reactive_ub"]}
                    ],
                    "minItems": 3, "maxItems": 3, "additionalItems": false
                },
                "values": {
                    "type": "array",
                    "items": [
                    {
                        "type": "array",
                        items": [
                            {"$ref": "#/values/extended_number"},
                            {"$ref": "#/values/extended_number"},
                            {"$ref": "#/values/extended_number"}
                        ],
                        "minItems": 3, "maxItems": 3, "additionalItems": false
                        }
                    ],
                    "minItemes": 1
                }
            }
        }
    }
}
```

In the generator GRG schema, `type` identifies the type of the object (i.e., a `generator`), `subtype` identifies the type of generator, `id` is a unique identifier, and `description` is an optional description field. `link` identifies the connecting network element (a node if the network is in node-breaker topology or bus-breaker topology, or a bus if the network is in a bus-branch topology). The `output` field defines the power S produced by the generator. Finally, the `PQ-curve` is a table of active and reactive bounds, defining the feasible operating region.

Table 14 summarizes generator components, and Figure 15 provides an example.

GRG name	symbol	unit
`output`→`active`	P	MegaWatt (*MW*)
`output`→`reactive`	Q	MegaVolt-Ampere Reactive (*MVAR*)
`PQ_curve`→`values`[k][0]	P_k	MegaWatt (*MW*)
`PQ_curve`→`values`[k][1]	Q_k^{min}	MegaVolt-Ampere Reactive (*MVAR*)
`PQ_curve`→`values`[k][2]	Q_k^{max}	MegaVolt-Ampere Reactive (*MVAR*)

Table 14: Generator: representation in the GRG format and units.

Synchronous Condenser

A synchronous condenser is a spinning machine that can compensate lagging current by either generating or absorbing reactive power.

Example: Generator

```
"generator_vl1_gn1" : {
    "type"   : "generator",
    "subtype": "solar",
    "id"     : "gn1",
    "link"   : "5",
        "PQ_curve" : {
           "arguments": ["active", "reactive_lb", "reactive_ub"],
           "values"   : [[0.0,  0.0, 0.0],
                         20.7, 0.0, 0.0]]
    }
    "output" : {
        "active"   : {"var" : {"lb" : 0.0, "ub" : 20.7}},
        "reactive" : {"var" : {"lb" : 0.0, "ub" : 0.0}}
    }
}
```

Figure 15: An example of a GRG generator.

GRG schema: synchronous_condenser

```
"synchronous_condenser": {
    "type"    : "object",
    "required": ["type", "id", "link", "output"],
    "additionalProperties": false,
    "properties": {
         "type"  : {"enum": ["synchronous_condenser"]},
         "id"    : {"type": "string"},
         "link"  : {"type": "string"},
         "output": {
             "reactive": {"$ref": "#/values/abstract_value"}
           }
    }
}
```

In the `synchronous_condenser` GRG schema, `type` identifies the type of the object (i.e., a `synchronous condenser`), the attribute `id` is a unique identifier, and `description` is an optional description field. `link` identifies the connecting network element (a node if the network is in node-breaker topology or bus-breaker topology, or a bus if the network is in a bus-branch topology). `output` defines the reactive power (`reactive`) produced or absorbed by the condenser.

Table 15 summarizes synchronous condenser components, and Figure 16 provides an example.

GRG name	symbol	unit
`output`→`reactive`	Q	MegaVolt-Ampere Reactive (*MVAR*)

Table 15: Synchronous condenser representation in GRG format and units.

Example: Synchronous Condenser

```
"synchronous_condenser_vl4_y1" : {
    "type"  : "synchronous_condenser",
    "id"    : "y1",
    "link"  : "2",
    "output": {
        "reactive" : {"var" : {"lb" : 0, "ub" : 14.53}}
    }
}
```

Figure 16: An example of a GRG synchronous condenser.

3 Assignment and Mappings

An *assignment* is a JSON object which contains a list of value assignments for some network component. Assignments may be used to define a particular instance of a network, for example, by assigning values to a set of variables. The assignment schema is as follows.

GRG schema: assignment

```
"assignment": {
    "type" : "object",
    "patternProperties": {
        "^(#|!|@)/.*": {
        "oneOf": [
            {"type": "object"},
            {"$ref": "#/abstract_value"},
            {"$ref": "#/abstract_status"}]
        }
    }
}
```

The JSON reference refers to any of the components described in section 2, and the `patternProperties` specifies that zero or more component assignments can be specified.

We say that a network component field is *assigned* if its value is specified as an object in an `assignment`. An assignment example is shown in Figure 17.

Example: Assignment

```
"assignment": {
    "#/network/components/substation_1/substation_components/voltage_level_1/
       voltage_level_components/switch_vl1_br1": {
        "status": "on"
     },
    "#/network/components/substation_1/substation_components/voltage_level_1/
      voltage_level_components/busbar_vl1_bbr1" : {
        "voltage": {
            "angle"    : 4.23,
            "magnitude": 63.12
        }
    },

}
```

Figure 17: An example of a GRG assignment.

A *mapping* is a set of assignments. It can be used to to define a particular instantiation of a network, or a desired network state, e.g., target values. Its schema is as follows.

GRG schema: mappings

```
"mappings": {
    "type": "object",
    "patternProperties": {
        "*": { "$ref": "#/assignment"}
    }
}
```

The `patternProperties` field specifies that zero or more set of component assignments can be specified. For each of these sets, an arbitrary number of component assignments can be specified. Figure 18 contains an example.

4 Networks

A GRG network is a collection of network components, along with a list of component assignments. When a network component is assigned, its value in the *assignments* block of the GRG document supersedes its value in the *network* block. The GRG schema for a network is shown below.

Example: Mappings

```
"mappings": {
    "target_points": {
        "#/network/components/substation_1/substation_components/voltage_level_1/
           voltage_level_components/generator_vl1_gn1": {
              "output" : {"active" : 0, "reactive" : 0}
         },
         "#/network/components/substation_1/substation_components/voltage_level_1/
           voltage_level_components/load_vl1_ld1": {
             "demand"  : {"active" : 4.1123, "reactive" : -1.8729}
         }
    },
    "starting_points": {
        "#/network/components/substation_1/substation_components/voltage_level_1/
           voltage_level_components/generator_vl1_gn1": {
              "output" : {"active" : 2.23, "reactive" : 0}
         },
         "#/network/components/substation_1/substation_components/voltage_level_1/
           voltage_level_components/load_vl1_ld1": {
             "demand"  : {"active" : 4.1123, "reactive" : -1.8729}
         }
    }
}
```

Figure 18: An example of GRG mappings.

GRG schema: network

```
"network": {
    "type": "object",
    "required": ["id", "type", "subtype", "per_unit", "components"],
    "properties": {
        "id": {"type": "string"},
        "type": {"enum": ["network"]},
        "subtype": {"enum": ["node_breaker", "bus_breaker", "bus_branch"]},
        "per_unit": {"type": "boolean"},
        "description": {"type": "string"},
        "components": {"$ref": "#/network/network_components"}
        "assignments": {"$ref": "#/assignments"},
    },
    "additionalProperties": false
}
```

The attribute `type` identifies the type of the object (i.e., a `network`). `subtype` indicates topology type, which defines the level of detail in descriptions of connections between components; `description` is an optional description field, and `per_unit` indicates whether network component values are expressed in per unit or nominal value.

A network is organized as a set of substations connected via transmission lines. Together these comprise the `network_components` block, as shown in the following schema fragment:

GRG schema: network components

```
"network_components": {
    "patternProperties": {
        ".*": {
            "oneOf": [
                {"$ref": "#/network_components/substation"},
                {"$ref": "#/network_components/ac_line"}
            ]
        }
    }
}
```

Substation

A *substation* is a collection of equipment located at a the same physical site and belonging to one Transmission System Operator (TSO). It is composed of several voltage levels and transformers. Its schema is as follows.

GRG schema: substation

```
"substation": {
    "type"    : "object",
    "required": ["type", "id", "substation_components"],

    "properties": {
        "type"       : {"enum": ["substation"]},
        "description": {"type": "string"},
        "id"         : {"type": "string"},
        "country"    : {"type": "string"},
        "TSO"        : {"type": "string"},
        "substation_components": {
            "patternProperties": {
                ".*": {
                    "oneOf": [
                        {"$ref": "#/network_components/voltage_level"},
                        {"$ref": "#/network_components/two_windings_transformer"}
                        {"$ref": "#/network_components/three_windings_transformer"}
                    ]
                }
            }
        }
    }
}
```

The attribute `type` identifies the type of the object (i.e., a `substation`). The attributes `country` and `TSO` indicate the country in which the substation is located, along with its Transmission System Operator. Substation components are listed in the creatively-named `substation_components` object.

Voltage Level

A *voltage level* is a collection of equipment located in the same substation at the same nominal voltage value. Its schema is as follows.

GRG schema: voltage_level

```
"voltage_level": {
    "type"    : "object",
    "required": ["type", "id", "voltage"],

    "properties": {
        "type"       : {"enum": ["voltage_level"]},
        "description": {"type": "string"},
        "id"         : {"type": "string"},
        "voltage": {
            "type"       : "object",
            "required"   : ["nominal_value", "upper_limit", "lower_limit"],
            "additionalProperties": false,
            "properties": {
                "nominal_value": {"type": "number"},
                "upper_limit"  : {"type": "number"},
                "lower_limit"  : {"type": "number"}
            }
        },
        "voltage_level_components": {
            "oneOf": [
                {"$ref": "#/network_components/bus"},
                {"$ref": "#/network_components/shunt"},
                {"$ref": "#/network_components/generator"},
                {"$ref": "#/network_components/synchronous_condenser"}
                {"$ref": "#/network_components/load"},
                {"$ref": "#/network_components/switch"}
            ]
        }
    }
}
```

The attribute `type` identifies the type of the object (i.e., a `voltage_level`). `voltage` contains the nominal voltage V_{nom} along with voltage limits $[V^L, V^U]$ for all components contained in the voltage level. The voltage level components are listed in `voltage_components`.

Table 16 describes the voltage level element.

GRG name	Parameter	unit
`voltage→nominal_value`	V_{nom}	KiloVolt (KV)
`voltage→lower_limit`	V^L	KiloVolt (KV)
`voltage→upper_limit`	V^U	KiloVolt (KV)

Table 16: Voltage Level description.

5 Market

A `market` GRG block attaches operational costs to a network component. A `cost` JSON block represents a cost model for a component.

voltage_level definition

```
"market": {
    "type"    : "object",
    "required": ["costs"],
    "properties": {
        "costs": {

            "patternProperties": {
                "#/.*": {
                "type": "object",
                "properties": {
                    "type": {"enum": ["polynomial"]},
                    "required": ["arguments", "coefficient"],
                    "arguments": {
                        "type"    : "array",
                        "items"   : {"$ref": "#/grg_pointer"},
                        "minItems": 1, "uniqueItems": true
                    },
                    "coefficients": {
                        "type" : "array":
                        "items": {
                            "type" : "array",
                            "items": {{"type": "#/abstract_value"}},
                            "minItems": 1
                        },
                        "minItems": 1
                    }
                }
            }
        }
    }
}
```

Each item in the `costs` block is named with a pointer to its associated network element. The arguments are pointers to network elements, and coefficient items define the associated cost polynomial. The i-th row of a `coefficients` array identifies the coefficients, $a_{n-1}, \ldots, a_0$, associated with the i-th x_i argument in the `arguments` array, generating the polynomial:

$$a_{n-1}\, x_i^{n-1} + \ldots + a_1\, x_i + a_0$$

6 Units

A `units` object details the units adopted for each physical quantity. These may differ from unit associated to the description of various components in the previous sections. The JSON schema of `units` is defined below.

GRG schema: units

```
    "units": {
        "type": "object",
        "required": ["voltage", "current", "angle", "reactive_power", "active_power",
                     "impedance","resistance", "reactance", "conductance", "susceptance", "time"],

        "properties": {
            "voltage": {"enum": ["volt", "kilo_volt", "mega_volt", "pu"]},
            "current": {"enum": ["ampere", "kilo_ampere", "mega_ampere", "pu"]},
            "angle"  : {"enum": ["degree", "radian"]},
            "active_power": {"enum": ["watt", "kilo_watt", "mega_watt", "pu"]},
            "reactive_power": {"enum": ["volt_ampere_reactive", "mega_volt_ampre_reactive", "pu"]},
            "impedance": {"enum": ["ohm", "pu"]},
            "resistance": {"enum": ["ohm", "pu"]},
            "reactance": {"enum": ["ohm", "pu"]},
            "conductance": {"enum": ["siemens", "pu"]},
            "susceptance": {"enum": ["siemens", "pu"]},
            "time": {"enum": ["seconds", "minutes", "hours", "days", "months", "years"]}
        }
    }
```

7 GRG Document

A *GRG document* is defined as a collection of JSON blocks, and includes a `network` block, a `mapping` block, a `market` block, and a `units` block. Its schema is as follows.

GRG schema: GRG document

```
{
    "type": "object",
    "required": ["grid_version", "network", "units", "market", "mappings"],
    "additionalProperties": false,

    "properties": {
        "grid_version": {"type": "string"},
        "description":  {"type": "string"},
        "network":      {"$ref": "#/network"},
        "mappings":     {"$ref": "#/mappings"},
        "market":       {"$ref": "#/market"},
        "units":        {"$ref": "#/units"}
    }
}
```

8 Network Topology

A network may be represented in one of three different topologies, from finer to coarser level of detail: node-breaker, bus-breaker, or bus-branch. Table 17 summarizes these representations, indicating which ones capture the voltage level topology (i.e., whether all the elements and their connections are physical ones) or not (i.e., merely logical connections); which switch types are captured, and the representation of buses. We will now discuss each representation in detail.

8.1 Node-Breaker Topology

A node-breaker topology contains the highest level of detail for a network. All components are physical elements, including busbar sections, breakers, and disconnectors. In this topology, a voltage level is a collection of network

	Node-breaker	Bus-breaker	Bus-branch
Topology	yes	yes	no
Breakers	yes	yes	no
Disconnectors	yes	no	no
Bus Type	busbar	logical bus	bus

Table 17: Network Topologies.

components and connection *nodes*. Nodes are logical connection points labeled with values in $\mathbb{N} \cup \{0\}$. Each component is directly connected to one node (if it is a bus, shunt, generator, or load), or to two nodes (if it is an AC line, transformer, or switch). The connection of a network component to a node is described in its `link` attribute (or `link_1` and `link_2` attributes for a connector). The set of nodes in a voltage level is described implicitly: It is the union of `link` values of all network components in that voltage level. Figure 19 (a) shows a voltage level in the node-breaker topology where breakers are illustrated as white squares, and disconnectors as pairs of white circles.

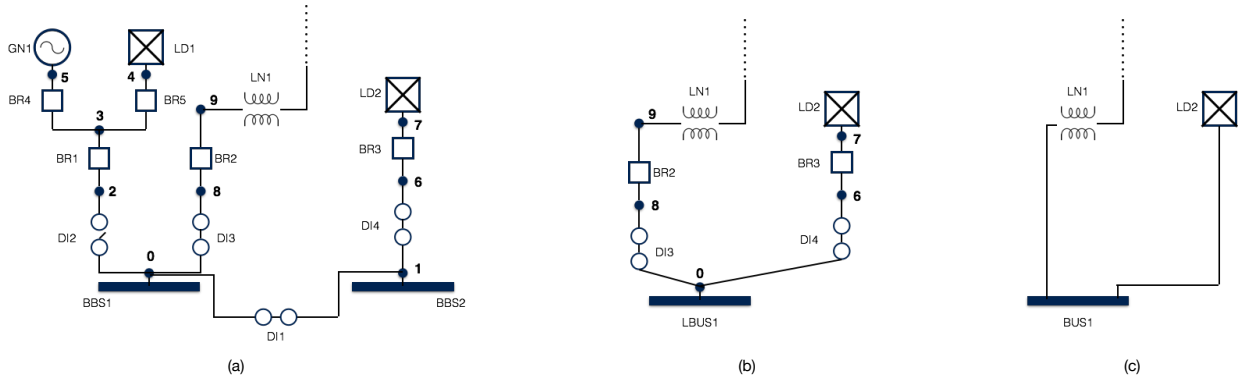

Figure 19: Illustration of a voltage level in node-breaker (a), bus-breaker (b), and bus-branch (c) topologies.

8.2 Bus-Breaker Topology

A network in bus-breaker form also represents a voltage level as a collection of components connected through nodes. However, the bus-breaker representation contains no disconnectors. Algorithm 1 describes the procedure for constructing a bus-breaker topology from a network in node-breaker form. The algorithm accepts a node-breaker network $\mathcal{N}$ as input. Lines (1–2) call the procedure Transform-VoltageLevel for each voltage level $\mathcal{V}$ in the network $\mathcal{N}$. The result of this operation is to (1) merge busbars of the voltage level which are connected by closed disconnectors, (2) remove all disconnectors from the voltage level, and (3) remove components that have no path to a logical bus. Lines (6–12) show greater detail for the removal process of a set of disconnectors $\mathcal{D}_\mathcal{V}$ of the voltage level $\mathcal{V}$. A disconnector is removed from the network by linking all voltage level components attached to its node on side 2 (`link_2`) to its node on side 1 (`link_1`) (lines 9–11). This operation may result in multiple busbars connected to the same node. In Figure 19 (a), for instance, busbars BBS1 and BBS2 will be connected to node 0 when disconnector DI1 is removed. The effect of the loop in lines 13–15 is to merge such busbar sets into single logical buses. This is done by preserving one busbar of the set B, changing its type to a logical bus, and removing all other busbars in B.

The effect of executing lines 16–18 is to remove all components of the voltage level for which there is no path through closed switches or non-assigned breakers to a logical bus of the voltage level. For instance, in Figure 19 (a), the breakers BR1, BR4, B5, the generator GN1, and the load LD1 have no closed path to a voltage level bus, since disconnector DI2 is open. Thus, these components are removed from the voltage level. The result of this operation is illustrated in Figure 19 (b).

Finally, in lines (3–5) of Algorithm 1, all lines and transformers (connectors) of the network are processed. As was done for voltage levels, a connector is removed if it has no closed path to logical buses on both sides.

Figure 19 (b) illustrates an example voltage level in the bus-breaker topology obtained from the node-breaker topology of Figure 19 (a).

Algorithm 1: NODE-BREAKER TO BUS-BREAKER($\mathcal{N}$)

1 **foreach** *voltage level* $\mathcal{V} \in \mathcal{N}$ **do**
2 Transform-VoltageLevel($\mathcal{V}$);
3 **foreach** *line and transformer* $l \in \mathcal{L}_{\mathcal{N}} \cup \mathcal{T}_{\mathcal{N}}$ **do**
4 **if** $\nexists$ *paths through close switches from* l *to some* $b_1 \in \mathcal{B}_{\mathcal{V}_1}$ *and* $b_2 \in \mathcal{B}_{\mathcal{V}_2}$ **then**
5 Remove l from $\mathcal{V}$;

Procedure Transform-VoltageLevel($\mathcal{V}$)

6 **foreach** *disconnectors* $d \in \mathcal{D}_{\mathcal{V}}$ **do**
7 $n_1 \leftarrow d[\textit{link\_1}]$;
8 $n_2 \leftarrow d[\textit{link\_2}]$;
9 **foreach** *component* $c \in \mathcal{V}$ *s.t.* c *is connected to* n_2 **do**
10 Let *link* be *link_i* if c is a *switch*, *line*, or *transformer*, and *link_i* $\neq n_2$;
11 $c[\textit{link}] \leftarrow n_1$;
12 Remove d from $\mathcal{V}$;
13 **foreach** *busbar* $b \in \mathcal{B}_{\mathcal{V}}$ **do**
14 Let $B = \{b' \in \mathcal{B}_{\mathcal{V}} \ : \ b'[\textit{link}] = b[\textit{link}]\}$;
15 Remove all $b' \in B \setminus \{b\}$;
16 **foreach** *component* $c \in \mathcal{V}$ **do**
17 **if** $\nexists$ *path through close switches from* c *to some* $b \in \mathcal{B}_{\mathcal{V}}$ **then**
18 Remove c from $\mathcal{V}$;

8.3 Bus-Branch Topology

A network in bus-branch form differs from one in a node- or bus-breaker form in that switches are not represented, and all network components are directly connected to a bus. Thus, the bus-branch topology has no concept of connection nodes.

A bus-branch topology can be obtained from a node- or a bus-breaker topology by applying a variation of Algorithm 1. Rather than processing disconnectors (line 6), all switches of the voltage level are processed. Finally, after the algorithm is executed, the `link` attribute of each generator, load, and shunt, and the `link_1` and `link_2` attributes of lines and transformers are updated to take the ID of the bus to which they are connected.

Figure 19 (c) illustrates an example voltage level in the bus-branch topology, obtained from the bus-breaker topology of Figure 19 (b).

9 Per-Unit Transformations

This section describes all per-unit transformations for network components.

AC Line

The per-unit transformation of an AC line (see also Figure 1) is shown in Table 18. The nominal current magnitude I_{nom} of the AC line is defined as:

$$I_{\text{nom}} = \frac{S_{\text{nom}}}{V_{\text{nom}}}, \tag{19}$$

where V_{nom} is voltage magnitude on either side of the line, and S_{nom} is the nominal power magnitude of the network. The nominal impedance of the AC line is defined as:

$$Z_{\text{nom}} = \frac{(V_{\text{nom}})^2}{S_{\text{nom}}} \tag{20}$$

GRG name	Parameter	Per-unit transformation
impedance→resistance	R	$\frac{1}{Z_{nom}} \cdot R$
impedance→reactance	X	$\frac{1}{Z_{nom}} \cdot X$
shunt_1→conductance	G_1	$Z_{nom} \cdot G_1$
shunt_1→susceptance	B_1	$Z_{nom} \cdot B_1$
shunt_2→conductance	G_2	$Z_{nom} \cdot G_2$
shunt_2→susceptance	B_2	$Z_{nom} \cdot B_2$
current_limits_1→values[k][1]	$S_{1_k}^{min}$	$\frac{1}{I_{nom}} \cdot I_{1_k}^{min}$
current_limits_1→values[k][2]	$S_{1_k}^{max}$	$\frac{1}{I_{nom}} \cdot I_{1_k}^{max}$
current_limits_2→values[k][1]	$S_{2_k}^{min}$	$\frac{1}{I_{nom}} \cdot I_{2_k}^{min}$
current_limits_2→values[k][2]	$S_{2_k}^{max}$	$\frac{1}{I_{nom}} \cdot I_{2_k}^{max}$
thermal_limits_1→values[k][1]	$S_{1_k}^{min}$	$\frac{1}{S_{nom}} \cdot S_{1_k}^{min}$
thermal_limits_1→values[k][2]	$S_{1_k}^{max}$	$\frac{1}{S_{nom}} \cdot S_{1_k}^{max}$
thermal_limits_2→values[k][1]	$S_{2_k}^{min}$	$\frac{1}{S_{nom}} \cdot S_{2_k}^{min}$
thermal_limits_2→values[k][2]	$S_{2_k}^{max}$	$\frac{1}{S_{nom}} \cdot S_{2_k}^{max}$

Table 18: AC line per-unit transformations.

Two windings transformer

For a two windings transformer (see Figure 3), the per-unit transformation is defined in Table 19. The nominal current magnitudes I_{nom}^1 on side 1 and I_{nom}^2 on side 2 are defined as:

$$I_{nom}^1 = \frac{S_{nom}}{V_{nom}^1} \tag{21}$$

$$I_{nom}^2 = \frac{S_{nom}}{V_{nom}^2}, \tag{22}$$

where V_{nom}^1 and V_{nom}^2 are the voltage magnitudes on sides 1 and 2 of the transformer. The nominal impedance magnitude on side 2 (low voltage side) of the transformer is defined as:

$$Z_{nom} = \frac{(V_{nom}^2)^2}{S_{nom}} \tag{23}$$

Three windings transformer

For a three windings transformer (see Figure 5), the per-unit transformation is defined in Table 20. The nominal impedance magnitude on side 1 (high voltage side) of the transformer is defined as:

$$Z_{nom} = \frac{(V_{nom}^1)^2}{S_{nom}}, \tag{24}$$

where V_{nom}^1, V_{nom}^2, and V_{nom}^3 are voltage magnitudes at sides 1, 2, and 3 of the transformer.

Bus

The per-unit transformation for a bus is defined in Table 21, where V_{nom} is voltage magnitude at the voltage level of the bus. If the network is in *flat* representation, then the voltage component of the bus is transformed in per-unit according to the description provided in the last three rows of Table 21.

Shunt

For a shunt (see Figure 10), the per-unit transformation is defined in Table 22. The nominal impedance of the AC line is defined as:

$$Z_{nom} = \frac{(V_{nom})^2}{S_{nom}} \tag{25}$$

GRG name	Parameter	Per-unit transformation
tap_changer→impedance→resistance	R_k	$\frac{1}{Z_{nom}} \cdot R_k$
tap_changer→impedance→reactance	X_k	$\frac{1}{Z_{nom}} \cdot X_k$
tap_changer→admittance→conductance	G_k	$Z_{nom} \cdot G_k$
tap_changer→admittance→susceptance	B_k	$Z_{nom} \cdot B_k$
tap_changer→tap_ratio	r_k	r_k
tap_changer→angle_shift	δ_k	$\frac{\pi}{180} \cdot \delta_k$
tap_changer→steps→values[k][1]→impedance→resistance	R_k	$\frac{1}{Z_{nom}} \cdot R_k$
tap_changer→steps→values[k][1]→impedance→reactance	X_k	$\frac{1}{Z_{nom}} \cdot X_k$
tap_changer→steps→values[k][2]→admittance→conductance	G_k	$Z_{nom} \cdot G_k$
tap_changer→steps→values[k][2]→admittance→susceptance	B_k	$Z_{nom} \cdot B_k$
tap_changer→steps→values[k][3]→tap_ratio	r_k	r_k
tap_changer→steps→values[k][4]→angle_shift	δ_k	$\frac{\pi}{180} \cdot \delta_k$
current_limits_1→values[k][1]	$I_{1_k}^{min}$	$\frac{1}{I_{nom}} \cdot I_{1_k}^{min}$
current_limits_1→values[k][2]	$I_{1_k}^{max}$	$\frac{1}{I_{nom}} \cdot I_{1_k}^{max}$
current_limits_2→values[k][1]	$I_{2_k}^{min}$	$\frac{1}{I_{nom}} \cdot I_{2_k}^{min}$
current_limits_2→values[k][2]	$I_{2_k}^{max}$	$\frac{1}{I_{nom}} \cdot I_{2_k}^{max}$
thermal_limits_1→values[k][1]	$S_{1_k}^{min}$	$\frac{1}{S_{nom}} \cdot S_{1_k}^{min}$
thermal_limits_1→values[k][2]	$S_{1_k}^{max}$	$\frac{1}{S_{nom}} \cdot S_{1_k}^{max}$
thermal_limits_2→values[k][1]	$S_{2_k}^{min}$	$\frac{1}{S_{nom}} \cdot S_{2_k}^{min}$
thermal_limits_2→values[k][2]	$S_{2_k}^{max}$	$\frac{1}{S_{nom}} \cdot S_{2_k}^{max}$

Table 19: Two windings transformer per-unit transformations.

Load

For a load, the per-unit transformation is defined in Table 23.

Generator

For a generator, the per-unit transformation is defined in Table 24.

Synchronous Condenser

For a synchronous condenser, the per-unit transformation is defined in Table 25.

Voltage Level

Each voltage component of a voltage level is transformed according to Table 26.

GRG name	Parameter	Per-unit transformation
impedance_1→resistance	R_1	$\frac{1}{Z_{nom}} \cdot R$
impedance_1→reactance	X_1	$\frac{1}{Z_{nom}} \cdot X$
shunt→conductance	G	$Z_{nom} \cdot G_1$
shunt→susceptance	B	$Z_{nom} \cdot B_1$
tap_changer→steps→impedance_2→resistance	R_k^2	$\frac{1}{Z_{nom}} \cdot R_k^2$
tap_changer→steps→impedance_2→reactance	X_k^2	$\frac{1}{Z_{nom}} \cdot X_k^2$
tap_changer→steps→impedance_3→resistance	R_k^3	$\frac{1}{Z_{nom}} \cdot R_k^3$
tap_changer→steps→impedance_3→reactance	X_k^3	$\frac{1}{Z_{nom}} \cdot X_k^3$
tap_changer→steps→tap_ratio_2	r_k^2	r_k^2
tap_changer→steps→angle_shift_2	δ_k^2	$\frac{\pi}{180} \cdot \delta_k^2$
tap_changer→steps→tap_ratio_3	r_k^3	r_k^3
tap_changer→steps→angle_shift_3	δ_k^3	$\frac{\pi}{180} \cdot \delta_k^3$
tap_changer→steps→values[k][1]→impedance_2→resistance	R_k^2	$\frac{1}{Z_{nom}} \cdot R_k^2$
tap_changer→steps→values[k][1]→impedance_2→reactance	X_k^2	$\frac{1}{Z_{nom}} \cdot X_k^2$
tap_changer→steps→values[k][2]→impedance_3→resistance	R_k^3	$\frac{1}{Z_{nom}} \cdot R_k^3$
tap_changer→steps→values[k][2]→impedance_3→reactance	X_k^3	$\frac{1}{Z_{nom}} \cdot X_k^3$
tap_changer→steps→values[k][3]→tap_ratio_2	r_k^2	r_k^2
tap_changer→steps→values[k][4]→angle_shift_2	δ_k^2	$\frac{\pi}{180} \cdot \delta_k^2$
tap_changer→steps→values[k][5]→tap_ratio_3	r_k^3	r_k^3
tap_changer→steps→values[k][6]→angle_shift_3	δ_k^3	$\frac{\pi}{180} \cdot \delta_k^3$
current_limits_1→values[k][1]	$I_{1_k}^{min}$	$\frac{1}{I_{nom}} \cdot I_{1_k}^{min}$
current_limits_1→values[k][2]	$I_{1_k}^{max}$	$\frac{1}{I_{nom}} \cdot I_{1_k}^{max}$
current_limits_2→values[k][1]	$I_{2_k}^{min}$	$\frac{1}{I_{nom}} \cdot I_{2_k}^{min}$
current_limits_2→values[k][2]	$I_{2_k}^{max}$	$\frac{1}{I_{nom}} \cdot I_{2_k}^{max}$
current_limits_3→values[k][1]	$I_{3_k}^{min}$	$\frac{1}{I_{nom}} \cdot I_{3_k}^{min}$
current_limits_3→values[k][2]	$I_{3_k}^{max}$	$\frac{1}{I_{nom}} \cdot I_{3_k}^{max}$
thermal_limits_1→values[k][1]	$S_{1_k}^{min}$	$\frac{1}{S_{nom}} \cdot S_{1_k}^{min}$
thermal_limits_1→values[k][2]	$S_{1_k}^{max}$	$\frac{1}{S_{nom}} \cdot S_{1_k}^{max}$
thermal_limits_2→values[k][1]	$S_{2_k}^{min}$	$\frac{1}{S_{nom}} \cdot S_{2_k}^{min}$
thermal_limits_2→values[k][2]	$S_{2_k}^{max}$	$\frac{1}{S_{nom}} \cdot S_{2_k}^{max}$
thermal_limits_3→values[k][1]	$S_{3_k}^{min}$	$\frac{1}{S_{nom}} \cdot S_{3_k}^{min}$
thermal_limits_3→values[k][2]	$S_{3_k}^{max}$	$\frac{1}{S_{nom}} \cdot S_{3_k}^{max}$

Table 20: Three windings transformer per-unit transformations.

GRG name	Parameter	Per-unit transformation
voltage→magnitude	v	$\frac{1}{V_{nom}} \cdot v$
voltage→angle→lb	θ^l	$\frac{\pi}{180} \cdot \theta^l$
voltage→angle→ub	θ^u	$\frac{\pi}{180} \cdot \theta^u$
voltage→nominal_value	V_{nom}	1
voltage→lower_limit	V^L	$\frac{1}{V_{nom}} \cdot V^L$
voltage→upper_limit	V^U	$\frac{1}{V_{nom}} \cdot V^U$

Table 21: Voltage component per-unit transformations.

GRG name	Parameter	Per-unit transformation
shunt→conductance	G	$\frac{1}{Z_{nom}} \cdot G$
shunt→susceptance	B	$\frac{1}{Z_{nom}} \cdot B$

Table 22: Shunt per-unit transformations.

GRG name	Parameter	Per-unit transformation
demand→active	P	$\frac{1}{S_{nom}} \cdot P$
demand→reactive	Q	$\frac{1}{S_{nom}} \cdot Q$

Table 23: Load per-unit transformations.

GRG name	Parameter	Per-unit transformation
output→active	P	$\frac{1}{S_{nom}} \cdot P$
output→reactive	Q	$\frac{1}{S_{nom}} \cdot Q$
PQ_curve→values[k][0]	P_k	$\frac{1}{S_{nom}} \cdot P_k$
PQ_curve→values[k][1]	Q_k^{min}	$\frac{1}{S_{nom}} \cdot Q_k^{min}$
PQ_curve→values[k][2]	Q_k^{max}	$\frac{1}{S_{nom}} \cdot Q_k^{max}$

Table 24: Generator per-unit transformations.

GRG name	Parameter	Per-unit transformation
output→reactive	Q	$\frac{1}{S_{nom}} \cdot Q$

Table 25: Synchronous condenser per-unit transformations.

GRG name	Parameter	Per-unit transformation
voltage→nominal_value	V_{nom}	1
voltage→lower_limit	V^L	$\frac{1}{V_{nom}} \cdot V^L$
voltage→upper_limit	V^U	$\frac{1}{V_{nom}} \cdot V^U$

Table 26: Voltage Level per-unit transformations.