modelica-json parser

Install on Windows:

- Create MODELICAPATH environment variable and set it as path to Modelica Buildings Library
- Install JRE and JDK, 64-bit
- Install Node.js.
- Install dependencies and compile Java files, run “InstallOnWindows.bat”
- Test installation, from the **`\modelica-json`** directory, run the parser on Command Prompt:

```
node app.js -f test\FromModelica\Modulation.mo
```

Install on Ubuntu:

- Set MODELICAPATH environment: `export MODELICAPATH=${MODELICAPATH}:/usr/local/Modelica/Library/`
- Install Java and node: `sudo apt-get install nodejs npm default-jdk`
- Install dependencies: `make install`
- Compile the Java files: `make compile`
- Run the test cases: `npm test`

Detailed documentation: https://lbl-srg.github.io/modelica-json/

modelica-json parser

Use the parser: `node app.js -f <path of the file to parse>`

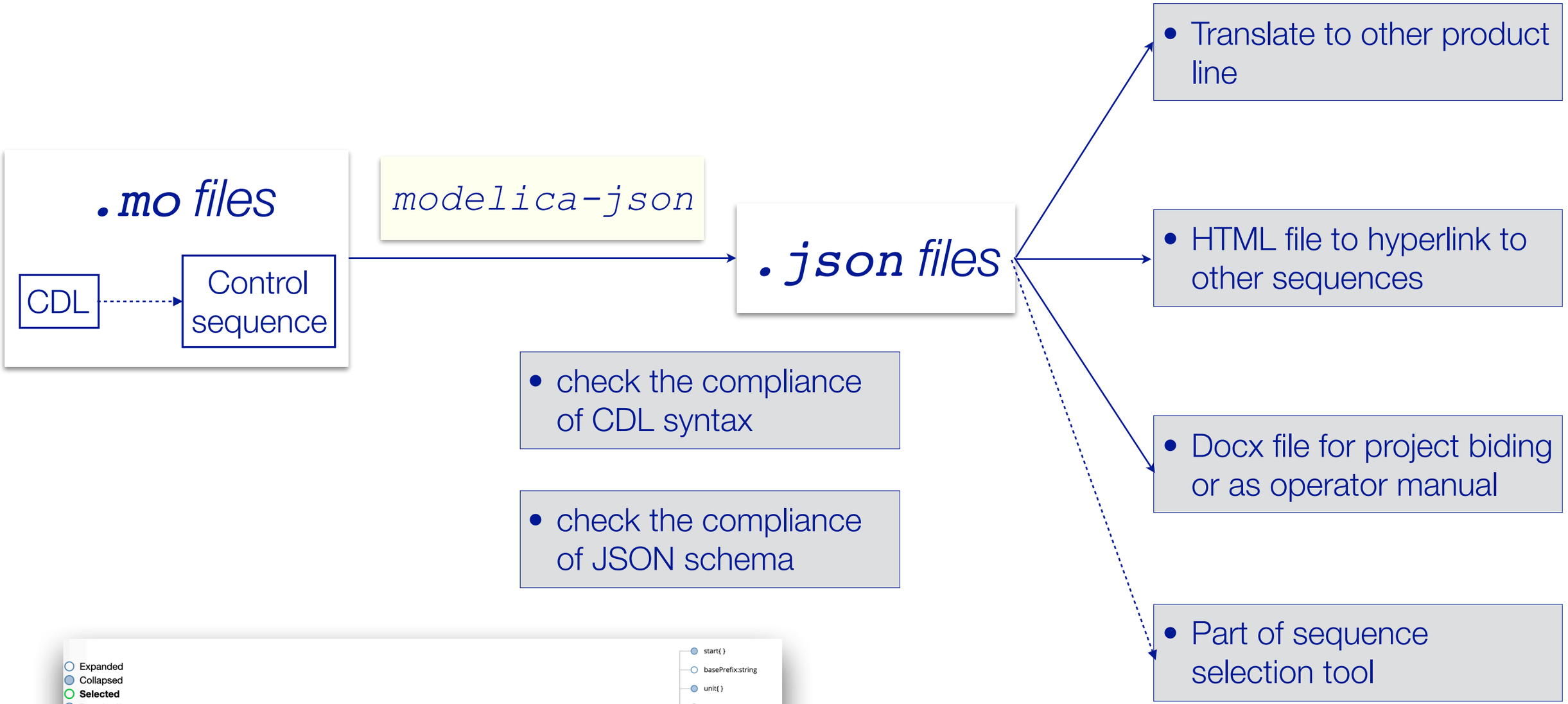
- `-file / -f`: path of the file or package to be parsed
- `-output / -o`: output format, '`raw-json`', '`json`' (default), '`html`', '`docx`'
- `-mode / -m`: translation mode, '`modelica`', '`cdl`' (default)
- `-log / -l`: logging level, '`error`', '`warn`', '`info`' (default), '`verbose`', '`debug`'
- `-directory / -d`: output directory, the default is the current directory

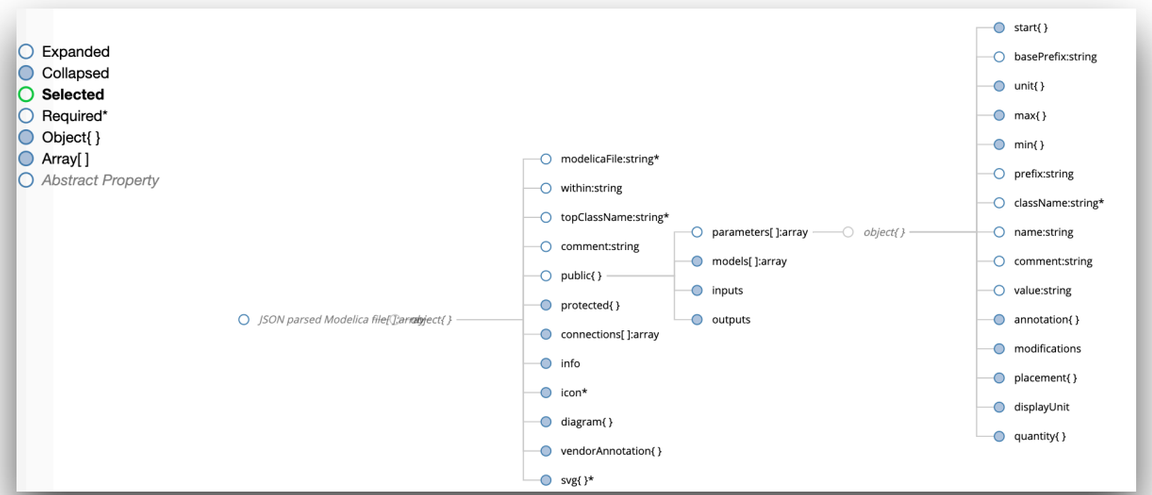
Validate JSON schemas: `node validate.js -f <path of the json file>`

Two schemas are available: `Schema-CDL.json`, `Schema-modelica.json`

Detailed documentation: https://lbl-srg.github.io/modelica-json/

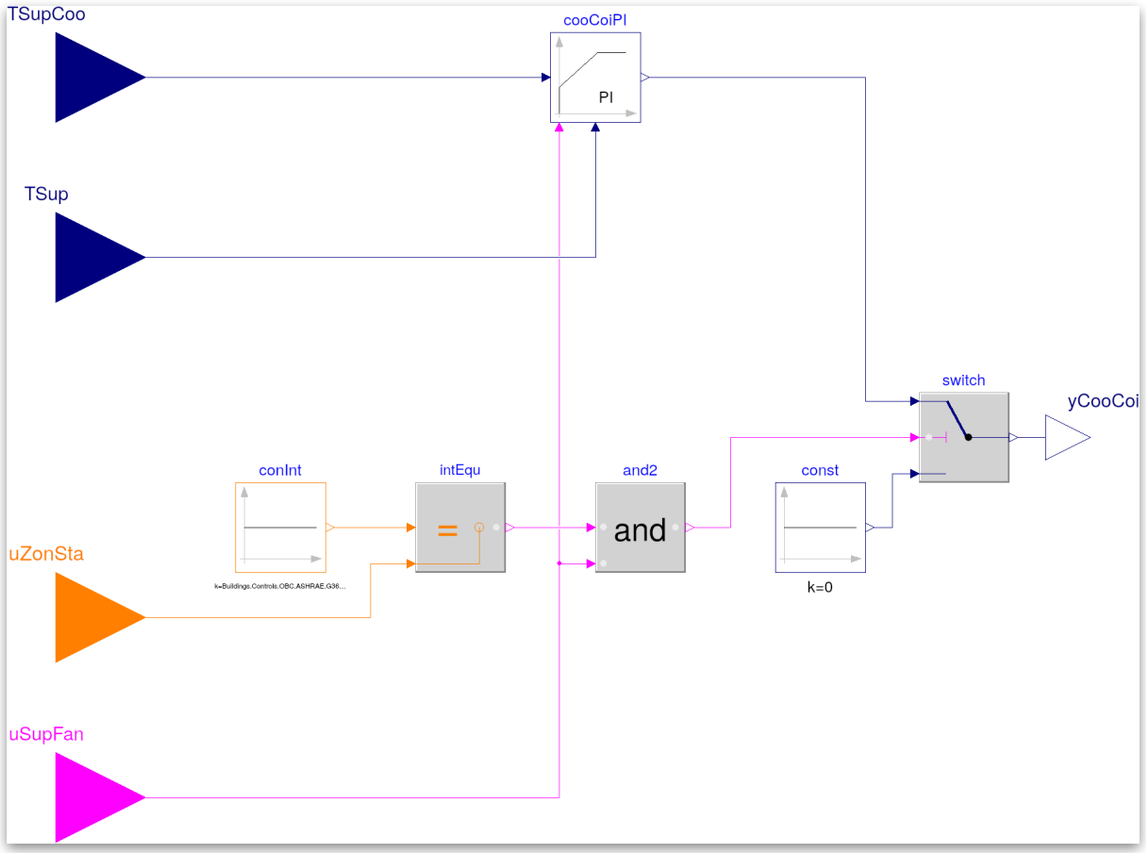
JSON file as intermedia format for documentations and further developments





Translate sequence with modelica-json: example

AHUs.SingleZone.VAV.CoolingCoil



"Output the cooling coil control signal (`yCooCoi`) if the fan is on (`uSupFan = true`) and the zone status (`uZonSta = cooling`) is in cooling mode. Otherwise, the control signal is set to `0`."

https://github.com/lbl-srg/modelica-json

```
within Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV;
model CoolingCoil "Controller for cooling coil valve"
  parameter Buildings.Controls.OBC.CDL.Types.SimpleController controllerTypeCooCoi=
    Buildings.Controls.OBC.CDL.Types.SimpleController.PI
    "Type of controller"
    annotation(Dialog(group="Cooling coil loop signal"));
……
  CDL.Continuous.LimPID cooCoiPI(
    reverseAction=true,
    reset=Buildings.Controls.OBC.CDL.Types.Reset.Parameter,
   …) "Cooling coil control signal"
    annotation (Placement(transformation(extent={{-10,70},{10,90}})));
……
 CDL.Interfaces.IntegerInput uZonSta "Zone state"
……
  CDL.Interfaces.RealOutput yCooCoi "Cooling coil control signal"
……
equation
  connect(const.y, switch.u3) annotation (Line(points={{62,-20},{66,-20},{66,-8},
          {70,-8}}, color={0,0,127}));
  connect(switch.u1, cooCoiPI.y)
    annotation (Line(points={{70,8},{60,8},{60,80},{12,80}},color={0,0,127}));
……
 annotation (defaultComponentName="cooCoi",
        Icon(coordinateSystem(preserveAspectRatio=false), graphics={}),
        Diagram(coordinateSystem(preserveAspectRatio=false)),
Documentation(info="<html>
<p>
This block outputs the cooling coil control signal if the fan is on and the zone
status
</p>
</html>",revisions="<html>
<ul><li>
August 1, 2019, by David Blum:<br/>
First implementation.
</li></ul>
</html>"));
end CoolingCoil;
```

From modelica to raw-json

```
node app.js -f models/CoolingCoil.mo -o raw-json
```

```
within Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV;
model CoolingCoil "Controller for cooling coil valve"
  parameter Buildings.Controls.OBC.CDL.Types.SimpleController controllerTypeCooCoi=
    Buildings.Controls.OBC.CDL.Types.SimpleController.PI
    "Type of controller"
    annotation(Dialog(group="Cooling coil loop signal"));
……
  CDL.Continuous.LimPID cooCoiPI(
    reverseAction=true,
    reset=Buildings.Controls.OBC.CDL.Types.Reset.Parameter,
   …) "Cooling coil control signal"
    annotation (Placement(transformation(extent={{-10,70},{10,90}})));
……
 CDL.Interfaces.IntegerInput uZonSta "Zone state"
……
  CDL.Interfaces.RealOutput yCooCoi "Cooling coil control signal"
……
equation
  connect(const.y, switch.u3) annotation (Line(points={{62,-20},{66,-20},{66,-8},
          {70,-8}}, color={0,0,127}));
  connect(switch.u1, cooCoiPI.y)
    annotation (Line(points={{70,8},{60,8},{60,80},{12,80}},color={0,0,127}));
……
 annotation (defaultComponentName="cooCoi",
        Icon(coordinateSystem(preserveAspectRatio=false), graphics={}),
        Diagram(coordinateSystem(preserveAspectRatio=false)),
Documentation(info="<html>
<p>
This block outputs the cooling coil control signal if the fan is on and the zone
status
</p>
</html>",revisions="<html>
<ul><li>
August 1, 2019, by David Blum:<br/>
First implementation.
</li></ul>
</html>"));
end CoolingCoil;
```


```
[
  {
    "within": [
      "Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV"
    ],
    "class_definition": [
      {
        "class_prefixes": "model",
        "class_specifier": {
          "long_class_specifier": {
            "name": "CoolingCoil",
            "comment": "\"Controller for cooling coil valve\"",
            "composition": {
              "element_list": {
                "element": [
                  {
                    "component_clause": {
                      "type_prefix": "parameter",
                      "type_specifier": "Buildings.Controls.OBC.CDL.Types.SimpleController",
                      "component_list": {
                        "component_declaration": [
                          {
                            "declaration": {
                              "name": "controllerTypeCooCoi",
                              "value": "Buildings.Controls.OBC.CDL.Types.SimpleController.PI"
                            },
                            "comment": {
                              "string_comment": "\"Type of controller\"",
                              "annotation": {
                                "dialog": [
                                  {
                                    "name": "group",
                                    "value": "\"Cooling coil loop signal\""
                                  }
                                ]
                              }
                            }
                          }
```

From modelica to json

```
node app.js -f models/CoolingCoil.mo -o json
```

```
within Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV;
model CoolingCoil "Controller for cooling coil valve"
  parameter Buildings.Controls.OBC.CDL.Types.SimpleController controllerTypeCooCoi=
    Buildings.Controls.OBC.CDL.Types.SimpleController.PI
    "Type of controller"
    annotation(Dialog(group="Cooling coil loop signal"));
……
  CDL.Continuous.LimPID cooCoiPI(
    reverseAction=true,
    reset=Buildings.Controls.OBC.CDL.Types.Reset.Parameter,
   …) "Cooling coil control signal"
    annotation (Placement(transformation(extent={{-10,70},{10,90}})));
……
 CDL.Interfaces.IntegerInput uZonSta "Zone state"
……
  CDL.Interfaces.RealOutput yCooCoi "Cooling coil control signal"
……
equation
  connect(const.y, switch.u3) annotation (Line(points={{62,-20},{66,-20},{66,-8},
          {70,-8}}, color={0,0,127}));
  connect(switch.u1, cooCoiPI.y)
    annotation (Line(points={{70,8},{60,8},{60,80},{12,80}},color={0,0,127}));
……
 annotation (defaultComponentName="cooCoi",
        Icon(coordinateSystem(preserveAspectRatio=false), graphics={}),
        Diagram(coordinateSystem(preserveAspectRatio=false)),
Documentation(info="<html>
<p>
This block outputs the cooling coil control signal if the fan is on and the zone
status
</p>
</html>",revisions="<html>
<ul><li>
August 1, 2019, by David Blum:<br/>
First implementation.
</li></ul>
</html>"));
end CoolingCoil;
```


```
[
 {
   "modelicaFile": "models/CoolingCoil.mo",
   "within": "Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV",
   "topClassName": "Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV.CoolingCoil",
   "comment": "Controller for cooling coil valve",
   "public": {
     "parameters": [
       {
         "className": "Buildings.Controls.OBC.CDL.Types.SimpleController",
         "type": "Buildings.Controls.OBC.CDL.Types.SimpleController",
         "name": "controllerTypeCooCoi",
         "value": "Buildings.Controls.OBC.CDL.Types.SimpleController.PI",
         "comment": "Type of controller",
         "unit": {
           "value": "\"1\""
         },
         "displayUnit": {
           "value": "\"1\""
         },
         "annotation": {
           "dialog": {
             "group": "Cooling coil loop signal",
             "tab": "General"
           }
         }
       },
……
```

From modelica to json

This process will:

- Simplify raw-json structure
- If there is composite block, parse recursively until only primitive CDL blocks
- Validate if the Modelica code has missing information.
- Validate the JSON representation against the CDL schema.

```
node app.js -f models/Parameter2.mo -o json -m cdl
```

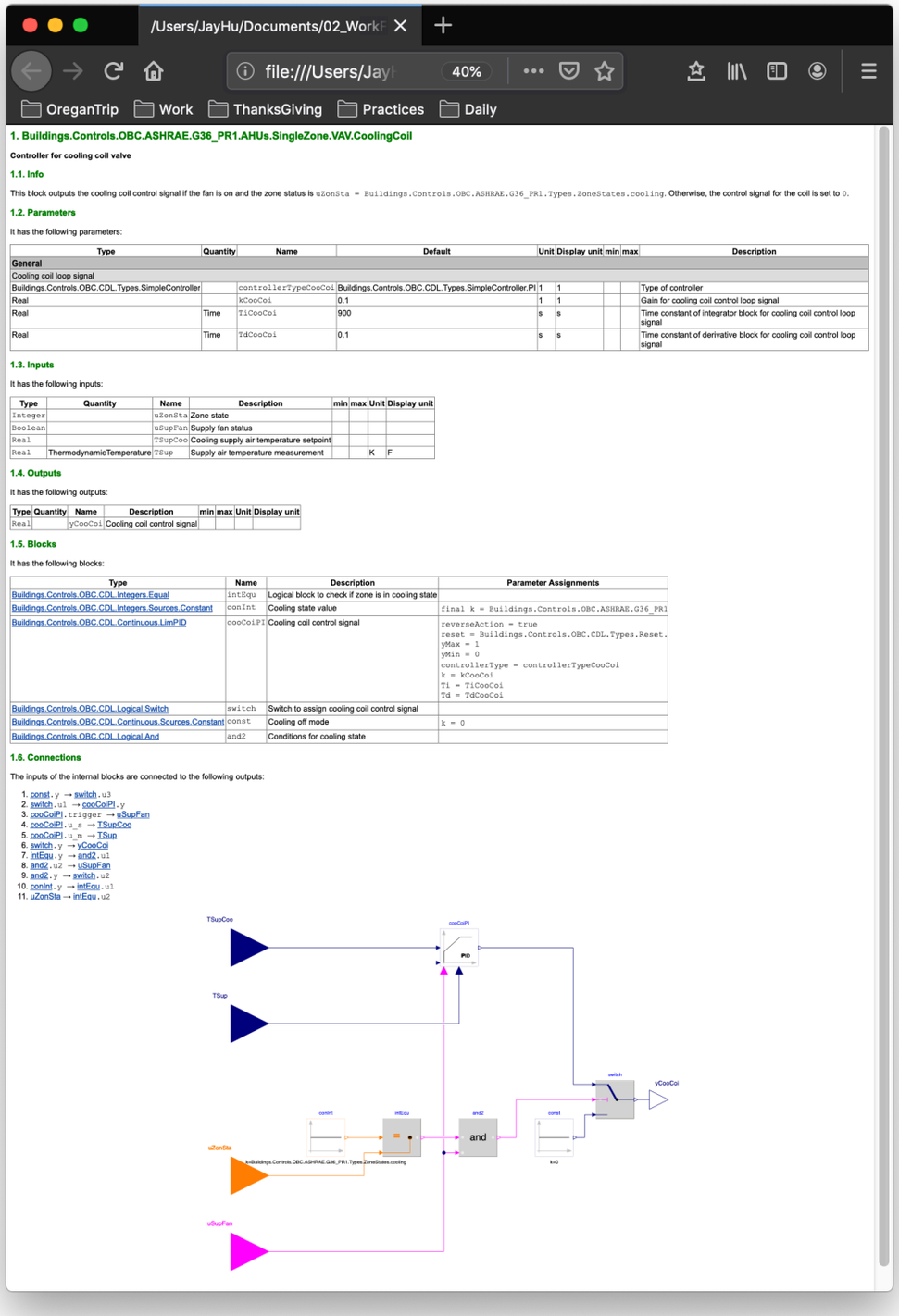
```
block Parameter2 "Some class comment"
  parameter Real myPar1 = 1;
  parameter Real myParNoValue "Some comment";
  parameter Real myParMin(min=0) "Some comment";
  parameter Real myParMax(max=0) "Some comment";
  parameter Real myParUnit(unit="K") "Some comment";

  parameter Real myParInGroup "Some comment"
     annotation(Dialog(group="Gains"));
  parameter Real myParInTab "Some comment"
     annotation(Dialog(tab="Initialization tab"));

  parameter Real myParInTabInGroup1 "Some comment 1"
     annotation(Dialog(tab="Initialization tab", group="Initial state"));
  parameter Real myParInTabInGroup2 "Some comment 2"
     annotation(Dialog(tab="Initialization tab", group="Initial state"));
end Parameter2;
```

```
warn:    Instance "myPar1" has no comment. Check Parameter2
warn:    Parameter2 has no info section.
```


```
Json file not valid, see errors below
data[0] should have required property 'icon', data[0] should have required property 'svg'
```

Need your contributions:

Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV.CoolingCoil



1. Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.SingleZone.VAV.CoolingCoil

Controller for cooling coil valve

1.1. Info

This block outputs the cooling coil control signal if the fan is on and the zone status is `uZonSta = Buildings.Controls.OBC.ASHRAE.G36_PR1.Types.ZoneStates.cooling`. Otherwise, the control signal for the coil is set to `0`.

1.2. Parameters

It has the following parameters:

Type	Quantity	Name	Default	Unit	Display unit	min	max	Description
General								
Cooling coil loop signal								
Buildings.Controls.OBC.CDL.Types.SimpleController		controllerTypeCooCoi	Buildings.Controls.OBC.CDL.Types.SimpleController.PI	1	1			Type of controller
Real		kCooCoi	0.1	1	1			Gain for cooling coil control loop signal
Real	Time	TiCooCoi	900	s	s			Time constant of integrator block for cooling coil control loop signal
Real	Time	TdCooCoi	0.1	s	s			Time constant of derivative block for cooling coil control loop signal

1.3. Inputs

It has the following inputs:

Type	Quantity	Name	Description	min	max	Unit	Display unit
Integer		uZonSta	Zone state				
Boolean		uSupFan	Supply fan status				
Real		TSupCoo	Cooling supply air temperature setpoint				
Real	ThermodynamicTemperature	TSup	Supply air temperature measurement			K	F

1.4. Outputs

It has the following outputs:

Type	Quantity	Name	Description	min	max	Unit	Display unit
Real		yCooCoi	Cooling coil control signal				

1.5. Blocks

It has the following blocks:

Type	Name	Description	Parameter Assignments
Buildings.Controls.OBC.CDL.Integers.Equal	intEqu	Logical block to check if zone is in cooling state	
Buildings.Controls.OBC.CDL.Integers.Sources.Constant	conInt	Cooling state value	`final k = Buildings.Controls.OBC.ASHRAE.G36_PR1`

Buildings.Controls.OBC.CDL.Continuous.LimPID	cooCoiPI	Cooling coil control signal	`reverseAction = true`
			`reset = Buildings.Controls.OBC.CDL.Types.Reset.`
			`yMax = 1`
			`yMin = 0`
			`controllerType = controllerTypeCooCoi`
			`k = kCooCoi`
			`Ti = TiCooCoi`
			`Td = TdCooCoi`

Buildings.Controls.OBC.CDL.Logical.Switch	switch	Switch to assign cooling coil control signal	
Buildings.Controls.OBC.CDL.Continuous.Sources.Constant	const	Cooling off mode	`k = 0`
Buildings.Controls.OBC.CDL.Logical.And	and2	Conditions for cooling state	

1.6. Connections

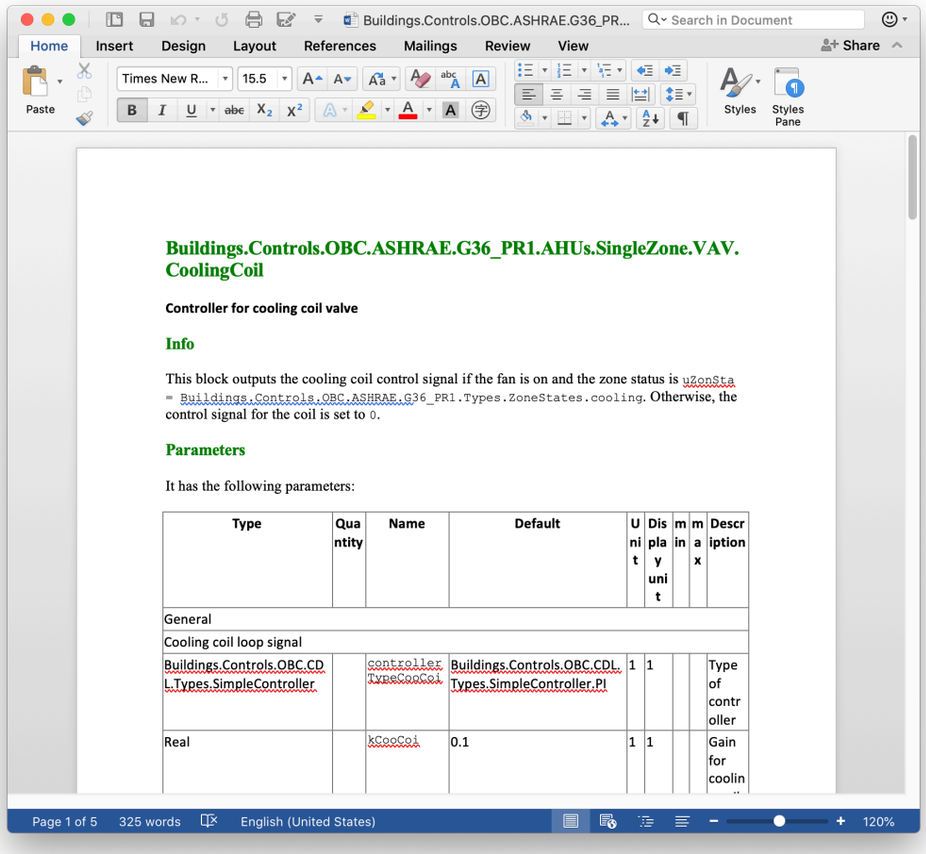
The inputs of the internal blocks are connected to the following outputs:

1. const.y → switch.u3
2. switch.u1 → cooCoiPI.y
3. cooCoiPI.trigger → uSupFan
4. cooCoiPI.u_s → TSupCoo
5. cooCoiPI.u_m → TSup
6. switch.y → yCooCoi
7. intEqu.y → and2.u1
8. and2.u2 → uSupFan
9. and2.y → switch.u2
10. conInt.y → intEqu.u1
11. uZonSta → intEqu.u2

Generate html:

```
node app.js -f models/CoolingCoil.mo -o html
```

- How to generate point list:
  - Could we have “Boolean” interfaces as DI/DO, and “Real/ Integer" Interfaces as AI/AO?
  - What the “Type" interface should be?
  - Do we need to specify hardware / software point type?
- How to layout the document
- Your convention of document the sequences
- How to export sequences for generating Brick model



Generate docx:

```
node app.js -f models/CoolingCoil.mo -o docx
```