OpenBuildingControl

Michael Wetter
Philip Haves
Jianjun Hu
Milica Grahovac

August 29, 2017

Lawrence Berkeley National Laboratory

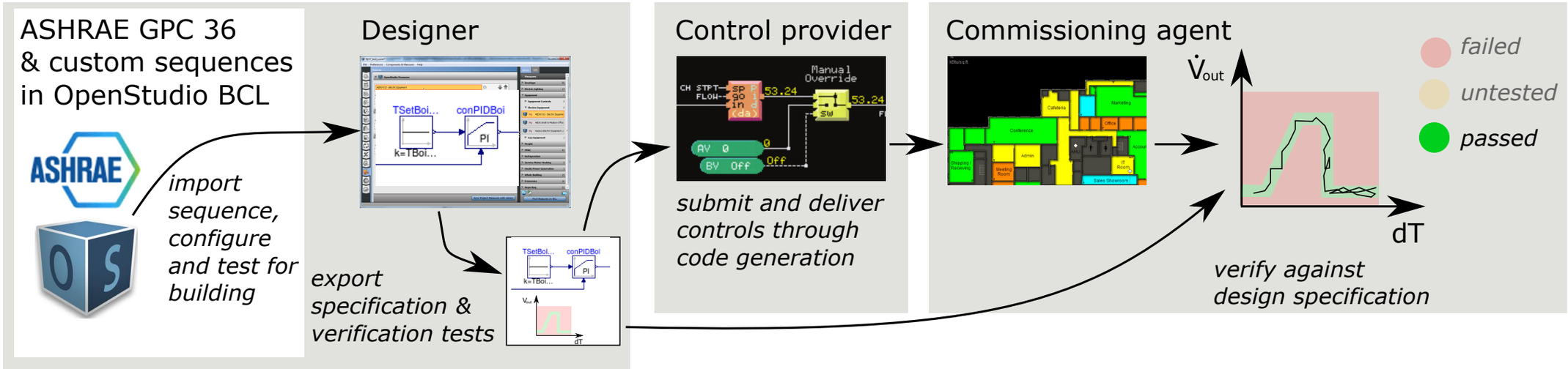
Presentation Contents

- OBC goals, objectives and approach
- Challenges
- Example control sequence representation
- Control description language (CDL)
- Questions and feedback

Goals and Objectives

- Develop a process and a set of tools to enable:
  - Design of, and sharing of, effective control sequences
  - Automated generation of vendor-specific code
  - Verification of correct implementation
  - Traceability from design to installation and operation, including changes
- Key characteristics:
  - Open standard
  - Open source
  - Vendor neutral
- Long term vision: automated control system design, including:
  - Component selection
  - Network configuration

OpenBuildingControl: Design and implement control sequences error-free and at lower cost to owner



Codify best practice | Design | Implement | Verify against original design intent

BACnet standardizes communication.

OpenBuildingControl will standardize
- basic functional building blocks that are used to compose sequences and tests,
- expressing control sequences,
- expressing functional verification tests,

for bidding, automatic implementation and automated functional testing.

Challenges

- Process
  - Define process based on viable business model(s)
- Technical
  - Control design tool:
    - Computational efficient modeling of actual control, local and supervisory, in a whole building simulation
    - Projects with no envelope + system model
  - Control description language:
    - Vendor-neutral but easily mapped to proprietary products
    - Extensible to support new technologies (e.g. MPC) and new control system architectures
- Industry
  - Engage critical mass of vendors, designers, contractors and owners to ensure wide adoption

Use cases and requirements

Various use cases are posted at http://obc.lbl.gov/specification/useCases.html

Requirements are posted at http://obc.lbl.gov/specification/requirements.html

All files can be edited on https://github.com/lbl-srg/obc/tree/master/specification/source

Sequence Specification

Implement atomic sequences using CDL

ASHRAE Guideline 36

cooling coil.

2. Supply Fan Speed Control and Supply Air Temperature Setpoint Reset

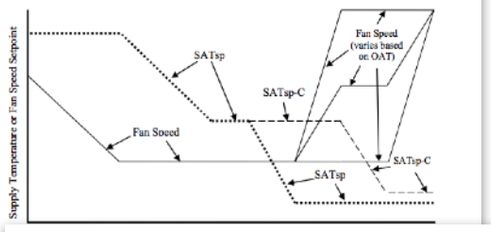
   a. The supply fan shall run whenever the unit is in any mode other than Unoccupied Mode.

   b. Provide a ramp function to prevent changes in fan speed of more than 10% per minute.

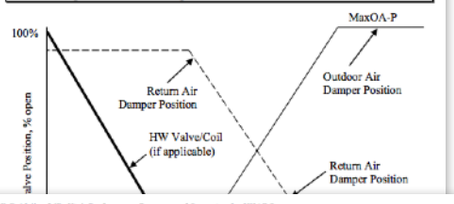
   c. Minimum, medium and maximum fan speeds shall be as follows:

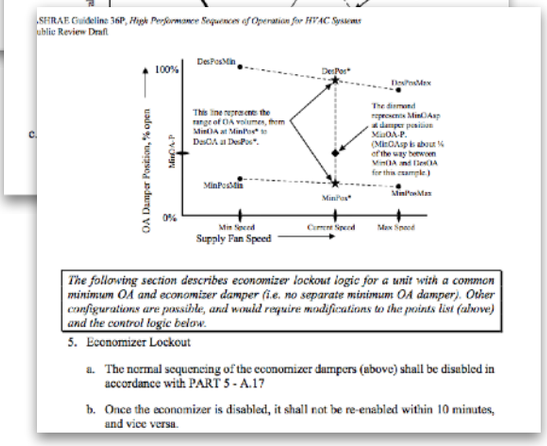
      1) Maximum cooling fan speed (MaxCoolSpeed), maximum heating fan speed (MaxHeatSpeed) and minimum fan speed (MinSpeed) setpoints shall be per 3.2B.1.

   e. When the supply fan is proven on, fan speed and supply air temperature setpoints are controlled as shown in the following diagrams and text. The points of transition along the x-axis shown and described below are representative. Separate gains shall be provided for each section of the control map (hot water, economizer, chilled water), that are determined by the Contractor to provide stable control. Alternatively, Contractor shall adjust the precise value of the x-axis thresholds shown in the figure to provide stable control.



actuators) and to maintain a more-linear relationship between fan speed and outdoor air volume.
In order to make this relationship as linear as possible, the economizer should use parallel blade dampers.





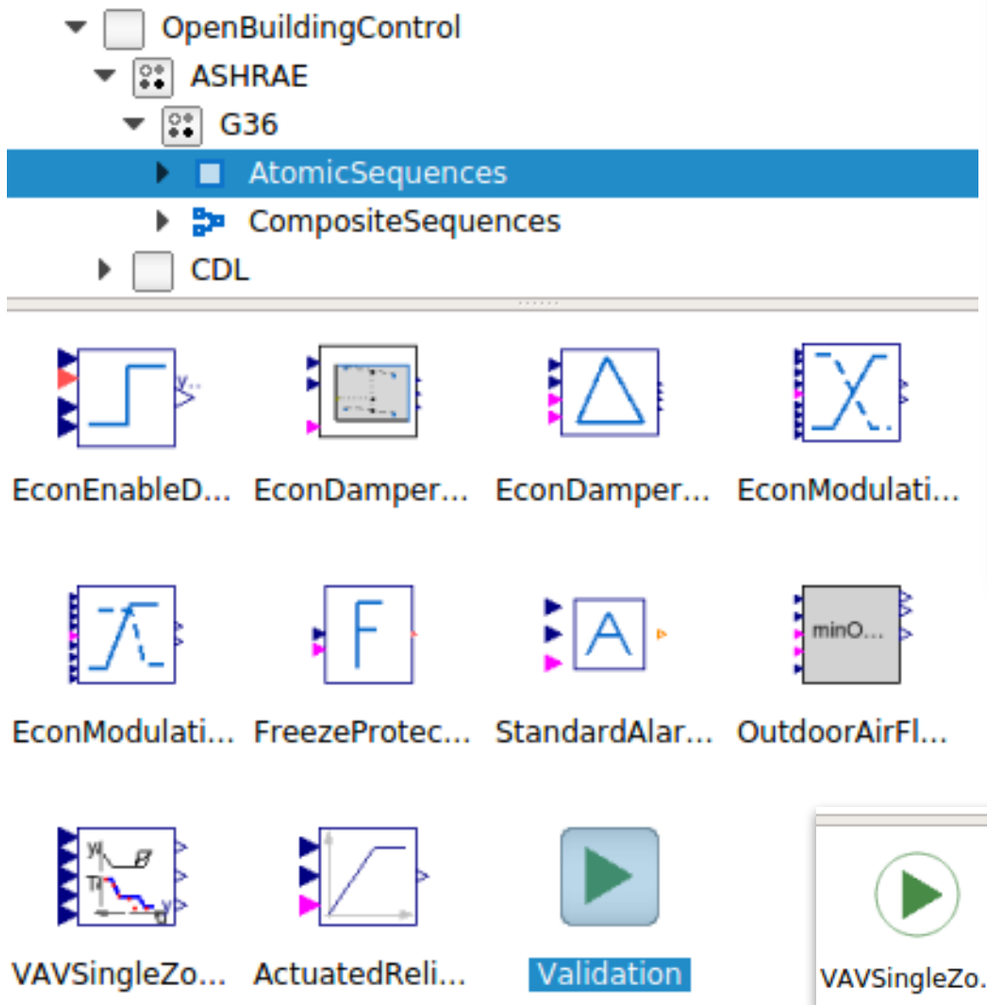
The following section describes economizer lockout logic for a unit with a common minimum OA and economizer damper (i.e. no separate minimum OA damper). Other configurations are possible, and would require modifications to the points list (above) and the control logic below.

5. Economizer Lockout

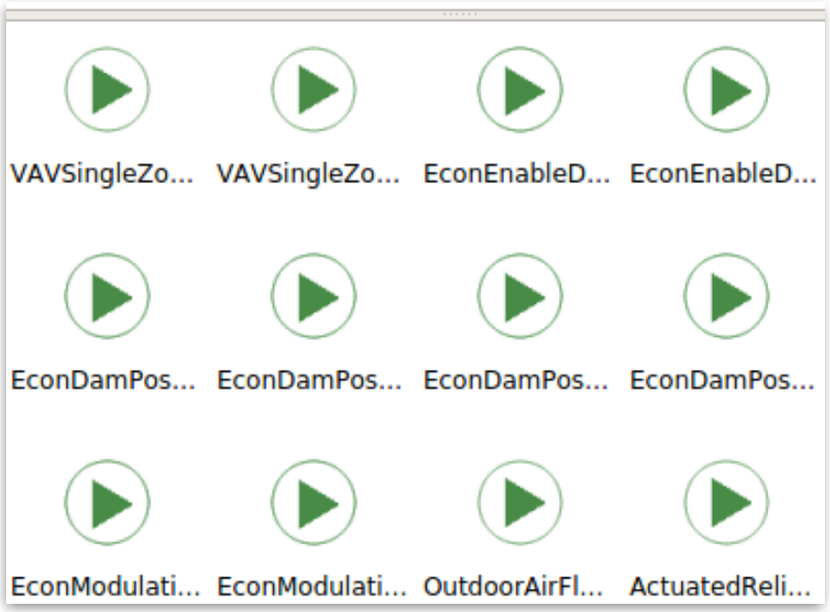
   a. The normal sequencing of the economizer dampers (above) shall be disabled in accordance with PART 5 - A.17

   b. Once the economizer is disabled, it shall not be re-enabled within 10 minutes, and vice versa.

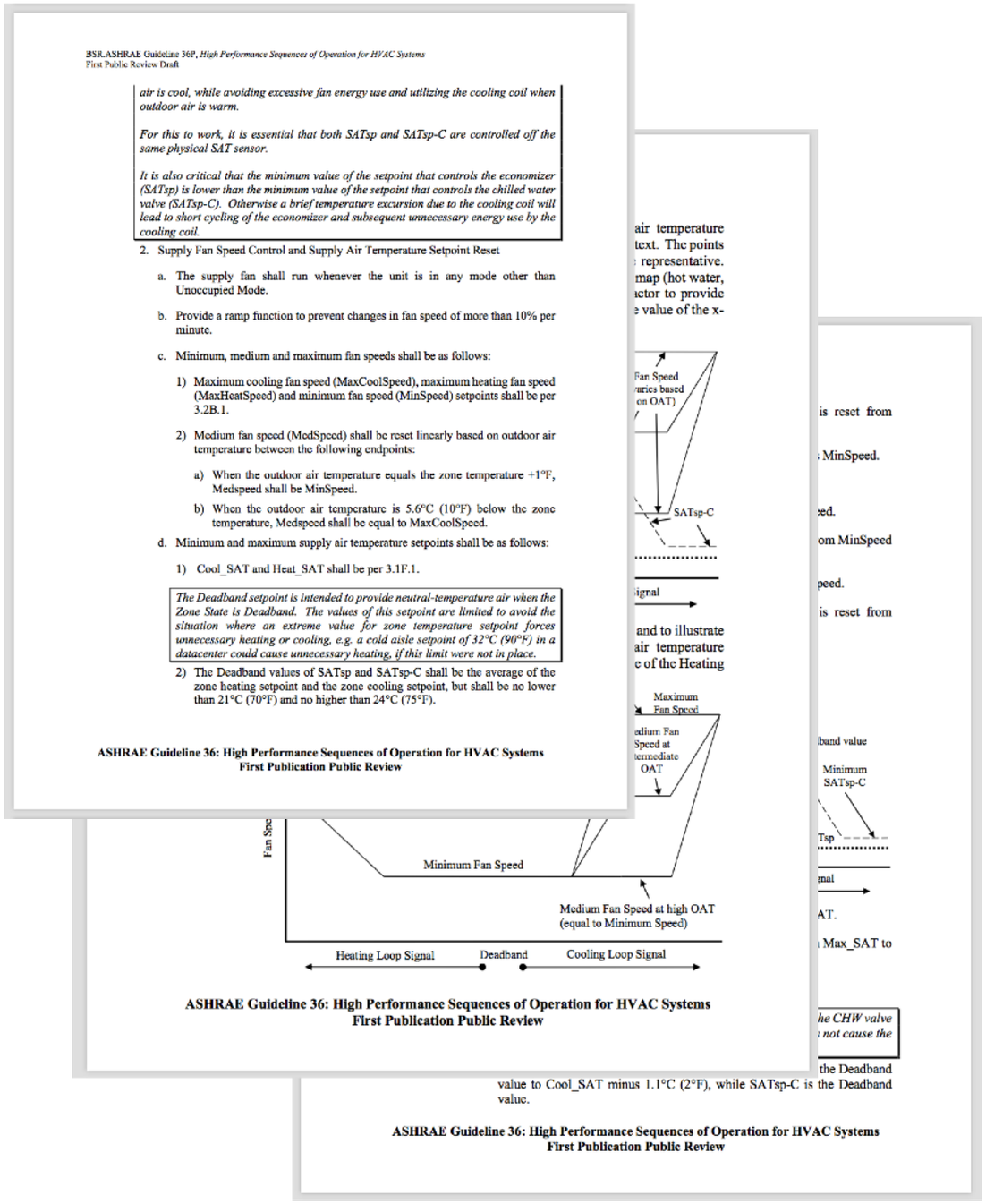
Implementation using CDL




```
CDL.Interfaces.BooleanInput uSupFan "Supply Fan Status, o
  a;
CDL.Interfaces.RealOutput yOutDamPos(min=0, max=1, unit="
                                                    a;
CDL.Interfaces.RealOutput yRetDamPos(min=0, max=1, unit="
                                                    a;
CDL.Continuous.Line outDamPos(limitBelow=true, limitAbove
  "Damper position is linearly proportional to the contro
  a;
CDL.Continuous.Line RetDamPos(limitBelow=true, limitAbove
  "Damper position is linearly proportional to the contro
  a;
CDL.Continuous.Constant minSignalLimit(k=damPosController
  "Identical to controller parameter - Lower limit of out
  a;
CDL.Continuous.Constant maxSignalLimit(k=damPosController
  "Identical to controller parameter - Upper limit of out
  a;
CDL.Interfaces.RealInput uHea(min=0, max=1, unit="1")
  "Heating control signal."
  a;
CDL.Interfaces.RealInput uCoo(min=0, max=1, unit="1")
  "Cooling control signal."
  a;
CDL.Interfaces.RealInput uOutDamPosMin(min=0, max=1, unit
  "Minimum economizer damper position limit as returned b
  a;
CDL.Interfaces.RealInput uOutDamPosMax(min=0, max=1, unit
  "Maximum economizer damper position limit as returned b
  a;
```



Implemented subset of ASHRAE Guideline 36 sequences using CDL



BSR/ASHRAE Guideline 36P, *High Performance Sequences of Operation for HVAC Systems*
First Public Review Draft

air is cool, while avoiding excessive fan energy use and utilizing the cooling coil when outdoor air is warm.

For this to work, it is essential that both SATsp and SATsp-C are controlled off the same physical SAT sensor.

It is also critical that the minimum value of the setpoint that controls the economizer (SATsp) is lower than the minimum value of the setpoint that controls the chilled water valve (SATsp-C). Otherwise a brief temperature excursion due to the cooling coil will lead to short cycling of the economizer and subsequent unnecessary energy use by the cooling coil.

2. Supply Fan Speed Control and Supply Air Temperature Setpoint Reset

   a. The supply fan shall run whenever the unit is in any mode other than Unoccupied Mode.

   b. Provide a ramp function to prevent changes in fan speed of more than 10% per minute.

   c. Minimum, medium and maximum fan speeds shall be as follows:

      1) Maximum cooling fan speed (MaxCoolSpeed), maximum heating fan speed (MaxHeatSpeed) and minimum fan speed (MinSpeed) setpoints shall be per 3.2B.1.

      2) Medium fan speed (ModSpeed) shall be reset linearly based on outdoor air temperature between the following endpoints:

         a) When the outdoor air temperature equals the zone temperature +1°F, Medspeed shall be MinSpeed.

         b) When the outdoor air temperature is 5.6°C (10°F) below the zone temperature, Medspeed shall be equal to MaxCoolSpeed.

   d. Minimum and maximum supply air temperature setpoints shall be as follows:

      1) Cool_SAT and Heat_SAT shall be per 3.1F.1.

The Deadband setpoint is intended to provide neutral-temperature air when the Zone State is Deadband. The values of this setpoint are limited to avoid the situation where an extreme value for zone temperature setpoint forces unnecessary heating or cooling, e.g. a cold aisle setpoint of 32°C (90°F) in a datacenter could cause unnecessary heating, if this limit were not in place.

      2) The Deadband values of SATsp and SATsp-C shall be the average of the zone heating setpoint and the zone cooling setpoint, but shall be no lower than 21°C (70°F) and no higher than 24°C (75°F).

ASHRAE Guideline 36: High Performance Sequences of Operation for HVAC Systems
First Publication Public Review

air temperature
text. The points
representative.
map (hot water,
actor to provide
value of the x-

Fan Speed
ries based
on OAT)

SATsp-C

is reset from

MinSpeed.

ed.

om MinSpeed

peed.

is reset from

and to illustrate
air temperature
e of the Heating

Maximum
Fan Speed

edium Fan
Speed at
termediate
OAT

band value

Minimum
SATsp-C

Tsp

gnal

AT.

Max_SAT to

Fan Speed

Minimum Fan Speed

Medium Fan Speed at high OAT
(equal to Minimum Speed)

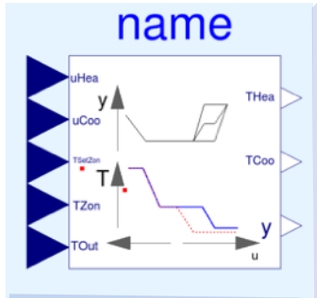
Heating Loop Signal | Deadband | Cooling Loop Signal

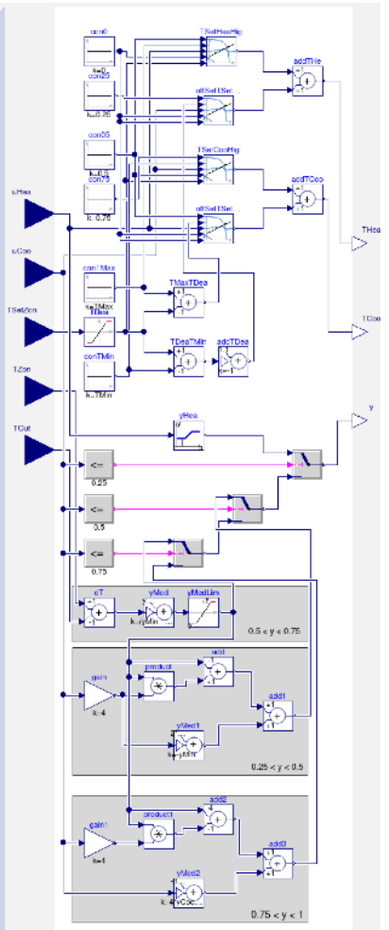
ASHRAE Guideline 36: High Performance Sequences of Operation for HVAC Systems
First Publication Public Review

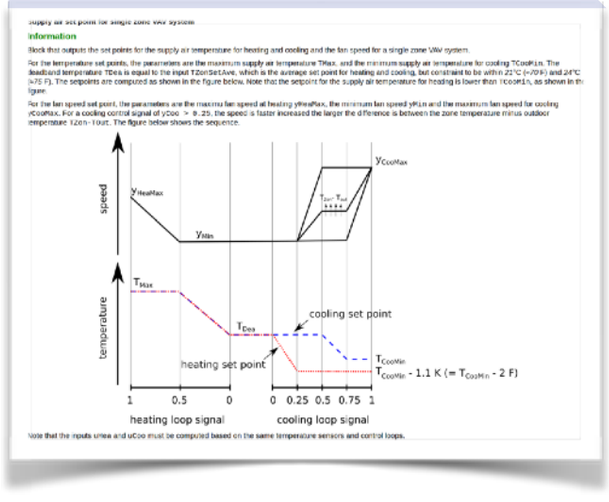
*he CHW valve
not cause the*

the Deadband
value to Cool_SAT minus 1.1°C (2°F), while SATsp-C is the Deadband value.

ASHRAE Guideline 36: High Performance Sequences of Operation for HVAC Systems
First Publication Public Review







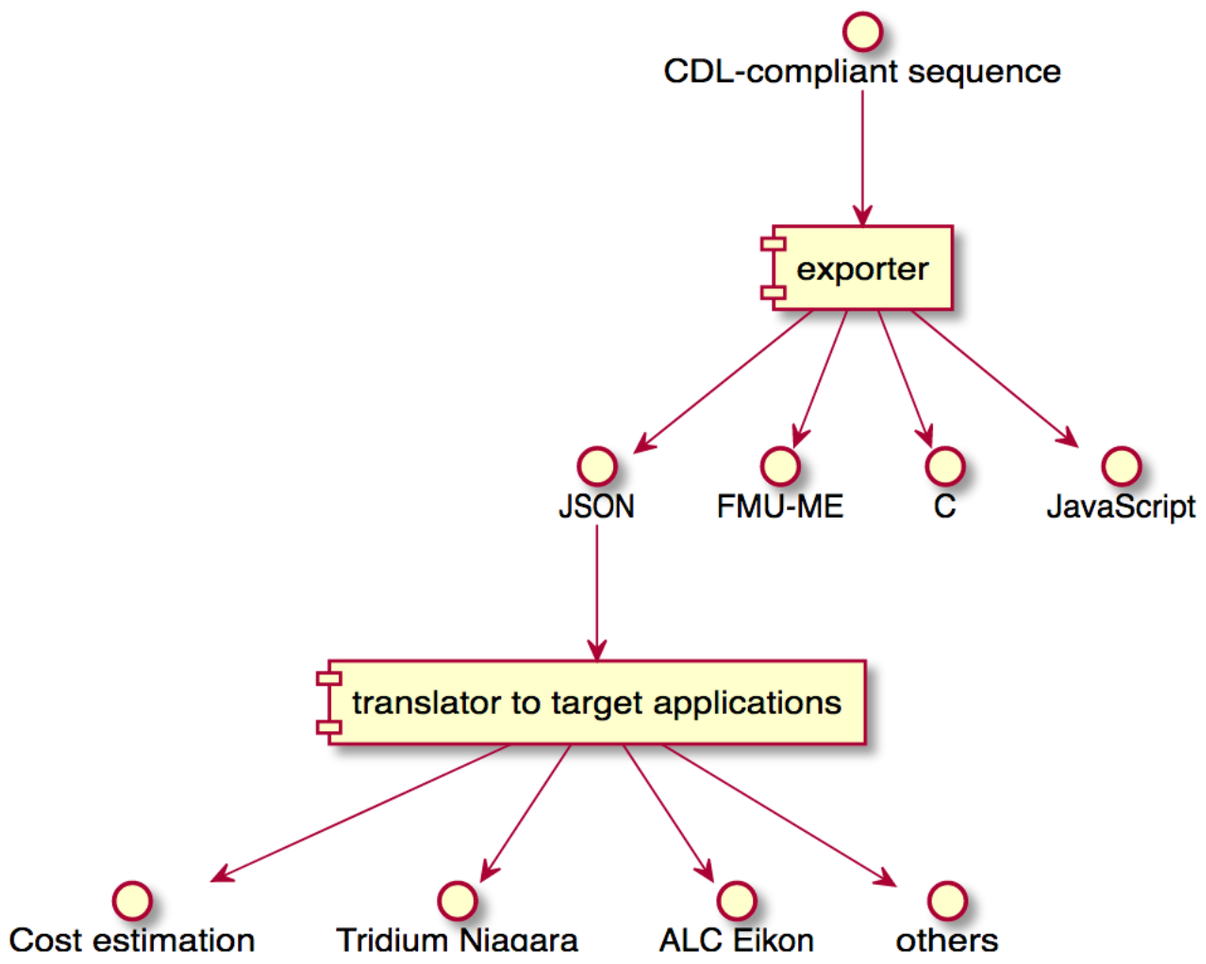
Control Description Language

What is CDL?

A language used to specify control sequences and verification tests.

Not a control sequence.

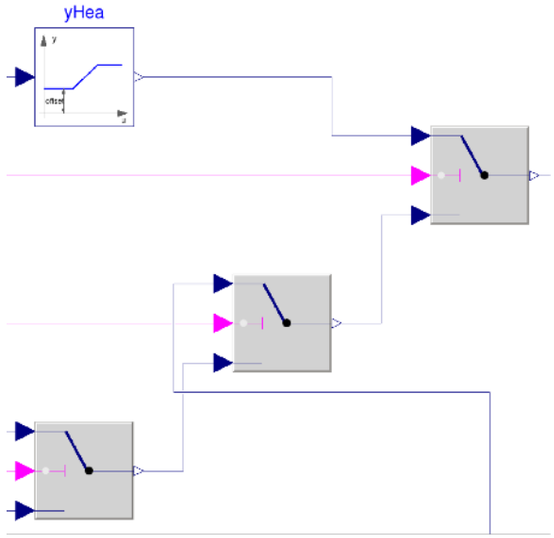
Control sequences are specified declaratively *using* CDL.



What is CDL?

A declarative language for expressing block-diagrams for controls (and requirements)

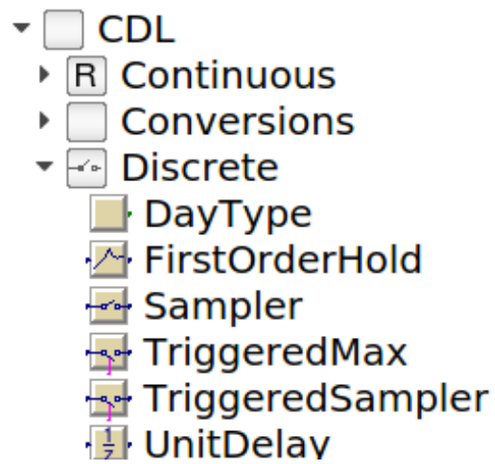
A graphical language for rendering these diagrams.



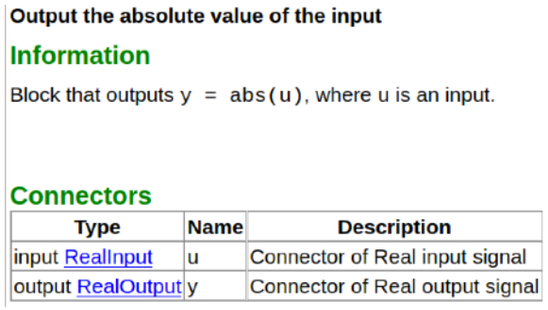
A library with elementary input/output blocks that should be supported [through a translator] by CDL-compliant control providers

Example: CDL has an adder with inputs `u1` and `u2`, gains `k1` and `k2`, and output

`y = k1*u1 + k2*u2`.



A syntax for documenting the control blocks and diagrams.



Output the absolute value of the input

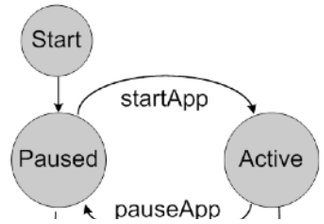
Information

Block that outputs `y = abs(u)`, where u is an input.

Connectors

Type	Name	Description
input RealInput	u	Connector of Real input signal
output RealOutput	y	Connector of Real output signal

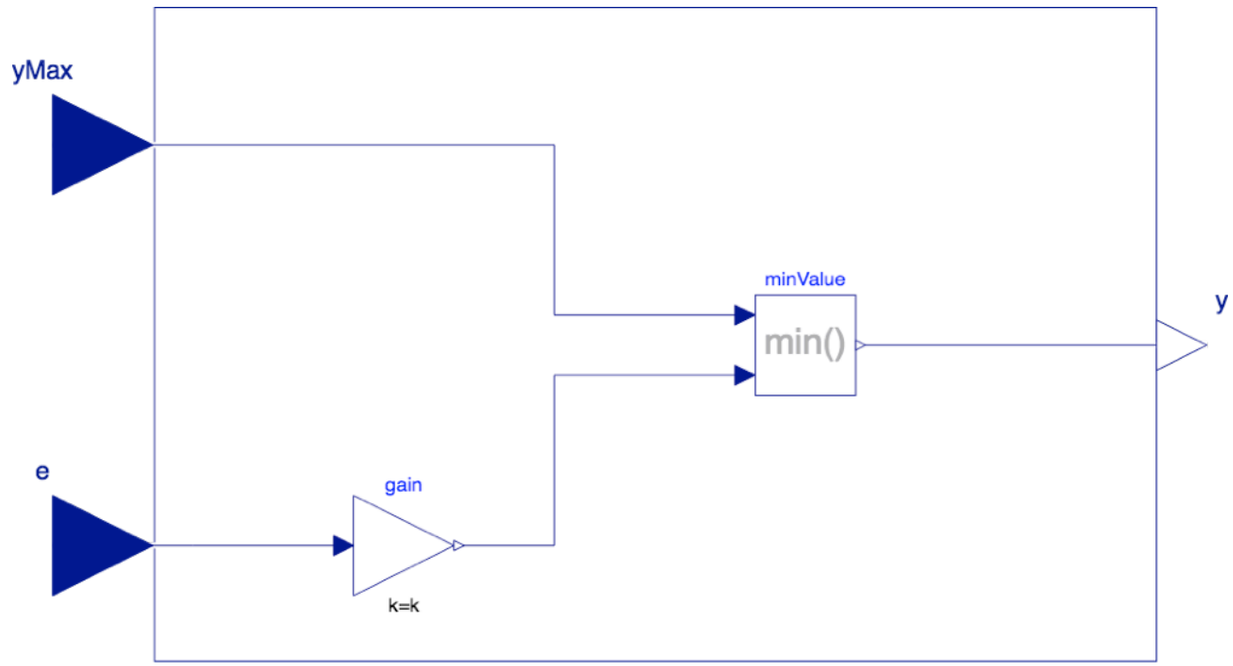
A model of computation that describes the interaction among the blocks.



Developed specification for review

Proposed

- Syntax
- Permissible data types
- Encapsulation of functionality
- Instantiation
- Connectors
- Connections
- Annotations
- Composite blocks
- Model of computations



See specification for details: http://obc.lbl.gov/specification/cdl.html

CDL library

Compared CDL library with industrial control library.

Validated blocks to ensure expected functionalities

- against known results
- across independent two independent simulators (Dymola and JModelica)

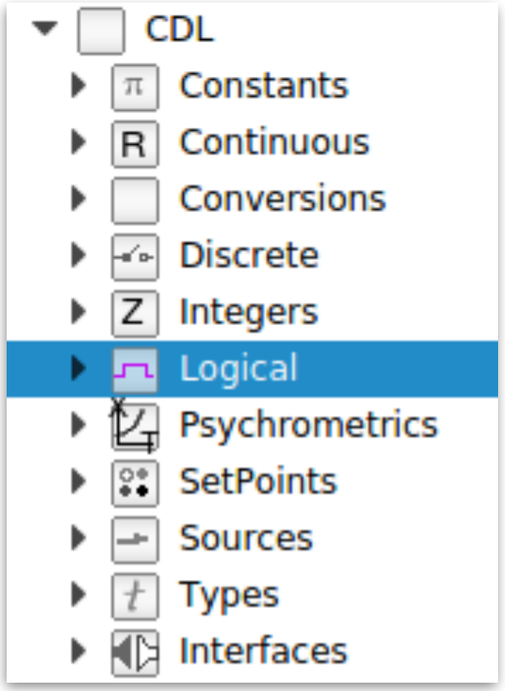
Package name	Description
Constants	Library of constants
Continuous	Library with elementary mathematical functions for continuous variables
Conversions	Library with blocks for type conversion
Discrete	Library of discrete input/output blocks with fixed sample period
Integers	Library with elementary mathematical functions for integer variables
Logical	Library with logical blocks
Psychrometrics	Library with psychrometric blocks
Routing	Package of blocks to combine and extract signals
SetPoints	Package with models for control set points
Types	Package with type definitions
Interfaces	Library with connectors for input and output signals

Browse CDL library at

http://obc.lbl.gov/specification/cdl/latest/help/CDL.html

In CDL library:

- 11 packages
- 117 basic blocks

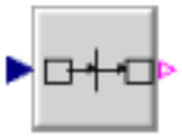


CDL library: Example

Some controls like window operation, economizer control, would need logic like:

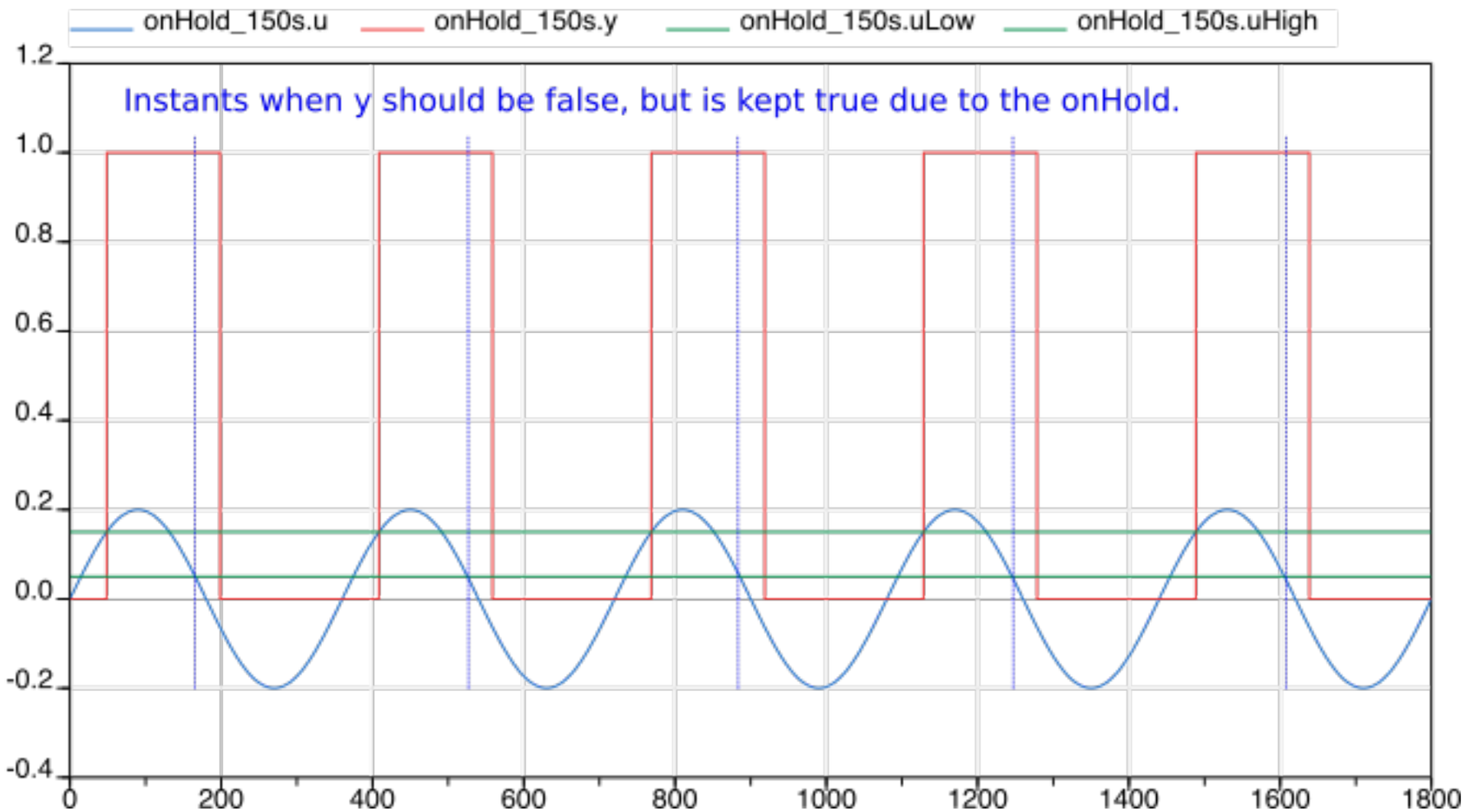
"If control input $u < 0.05$, output y switches to false; if control input $u > 0.15$, output y switches to true; the true output should not be changed for 150 seconds, regardless of the input change."

— need a control block with "hold" functionality



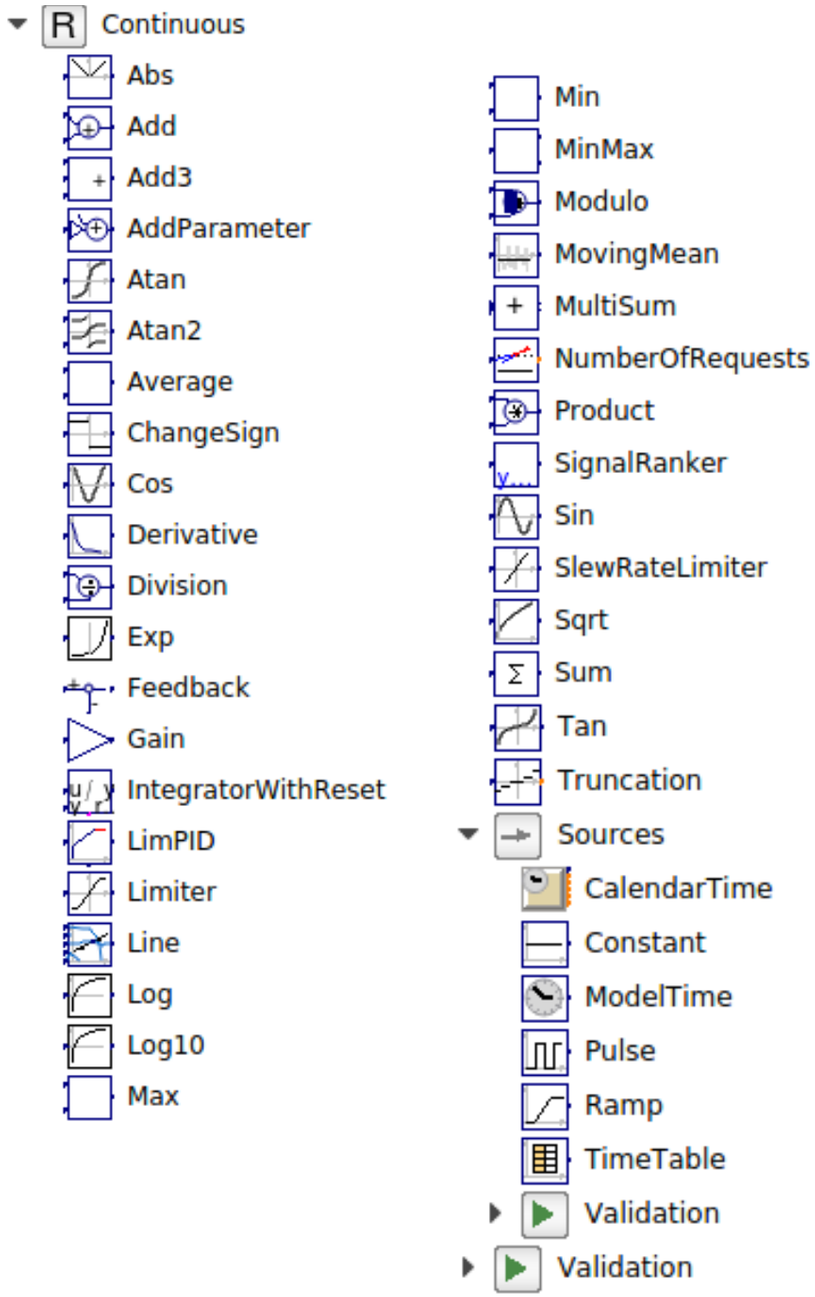
"HysteresisWithHold":

— allows to specify a hold time. During the hold time, the output is not allowed to switch.

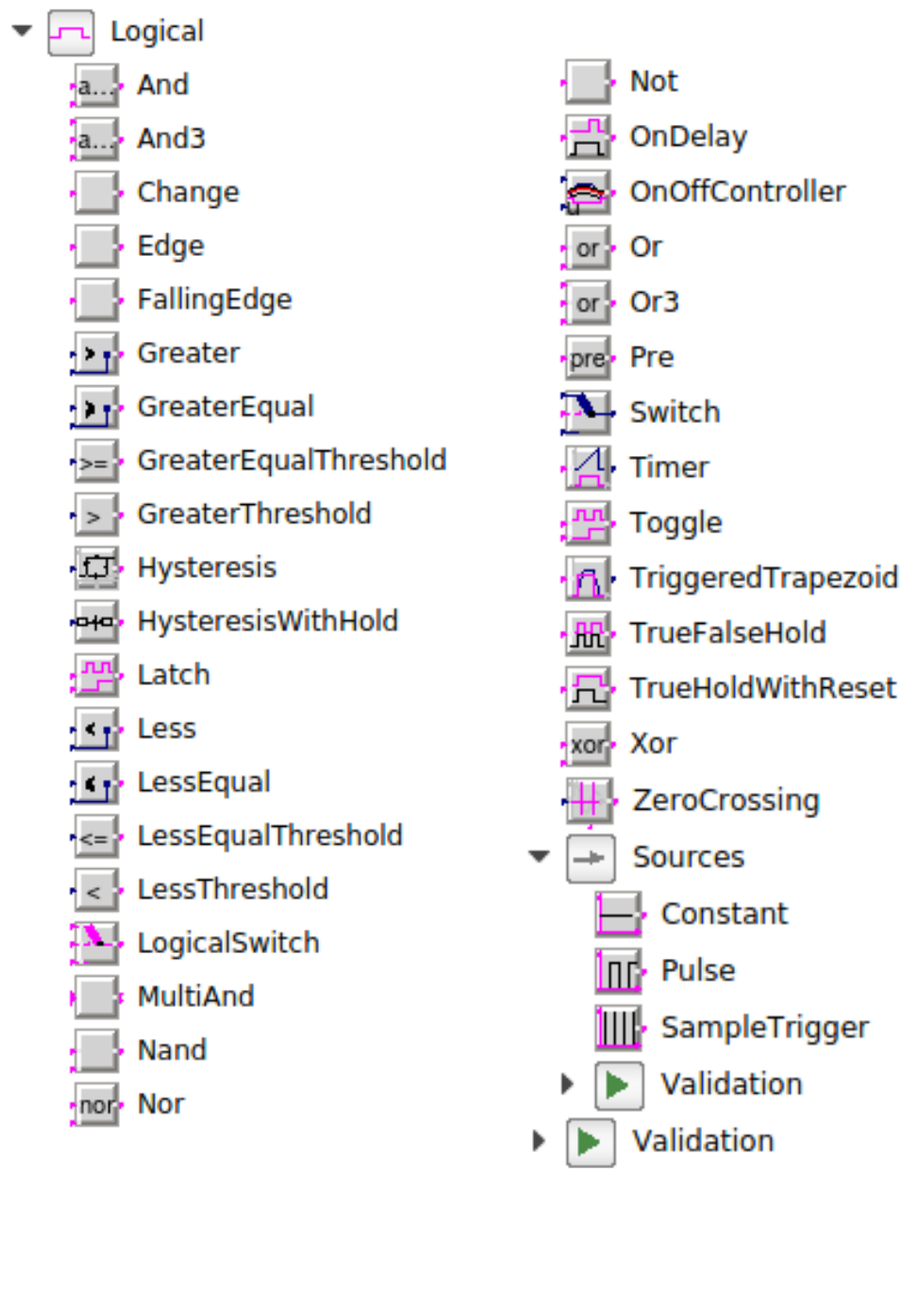


CDL library: Packages

`CDL.Continuous`:
elementary mathematical functions for continuous variables

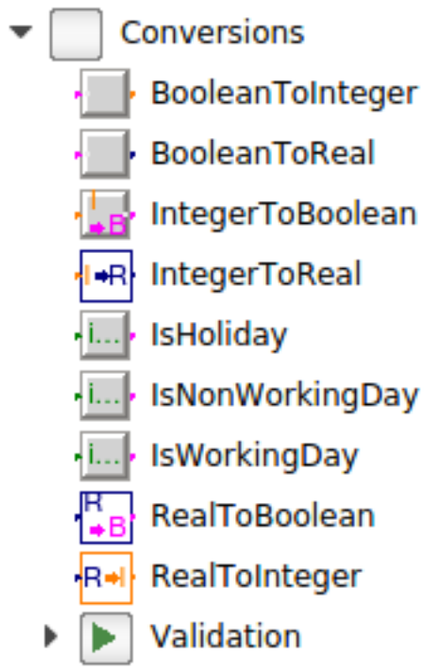


`CDL.Logical`:
elementary mathematical functions for boolean variables

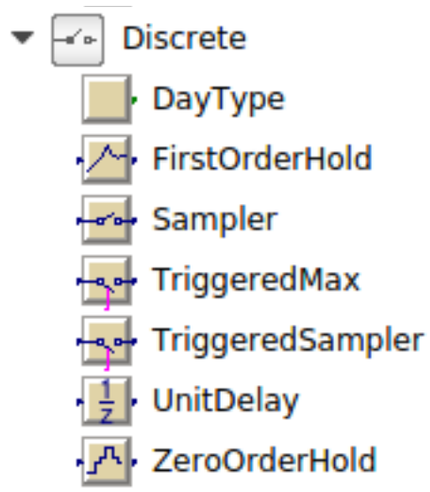


CDL library: Packages

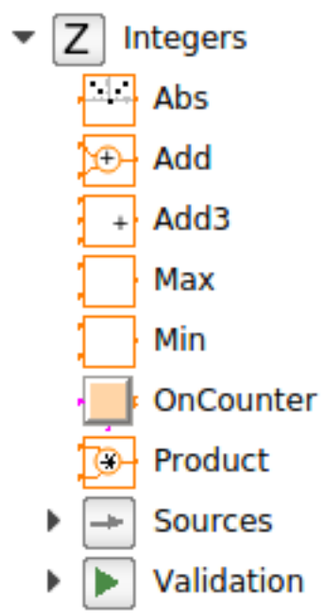
`CDL.Conversions`:
type conversions



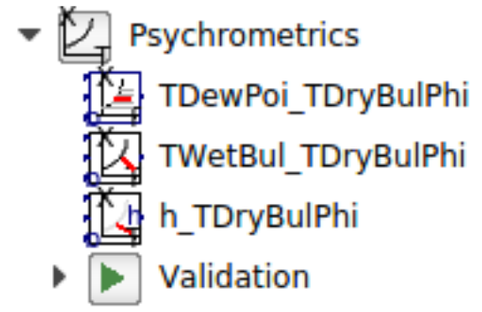
`CDL.Discrete`:
daytype, sample, delay, hold



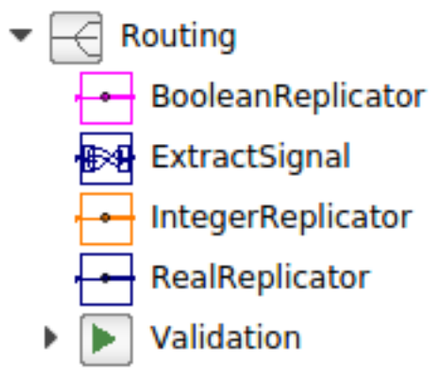
`CDL.Integers`:
mathematical functions for integer variables



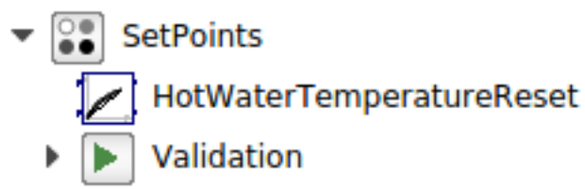
`CDL.Psychrometrics`:
psychrometric calculations



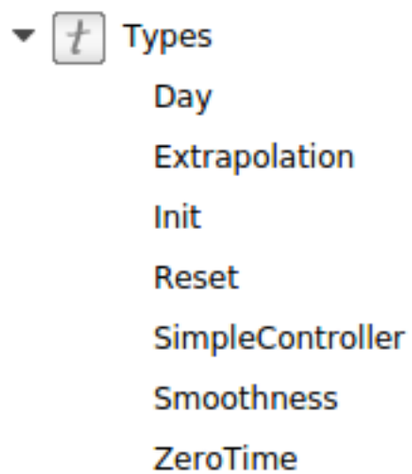
`CDL.Routing`:
combine and extract signals



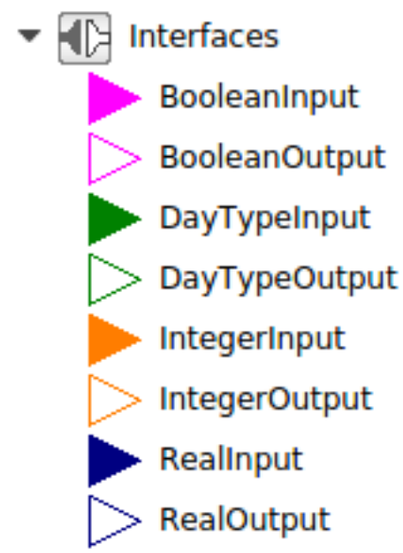
`CDL.Setpoints`:
setpoints for control systems



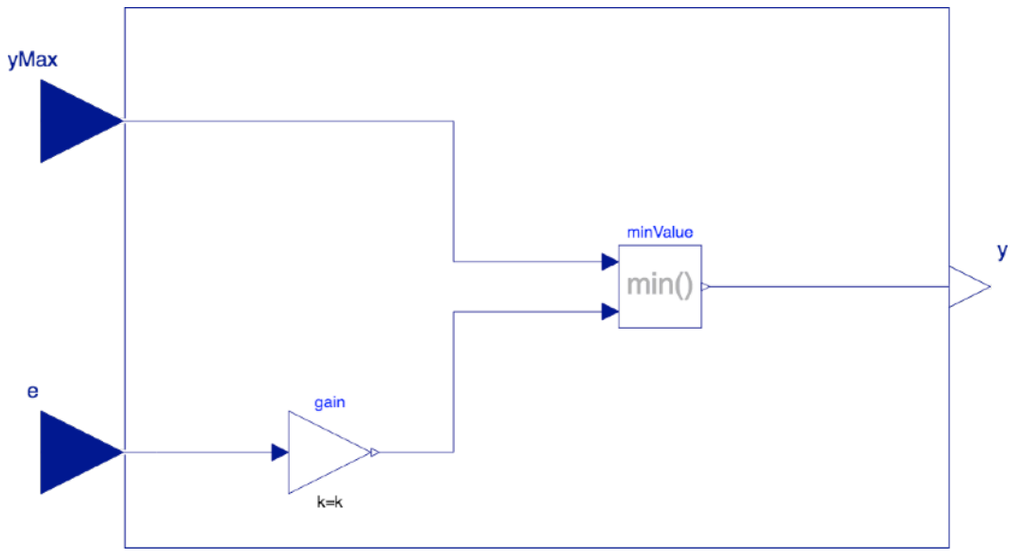
`CDL.Types`:
type definitions



`CDL.Interfaces`:
connectors for input and output signals



Custom sequences can be specified using blocks from CDL, pre-configure ASHRAE G36 sequences (and any custom-library that is based on CDL)




```
model CustomPWithLimiter
  "Custom implementation of a P controller with variable output
limiter"

  parameter Real k "Constant gain";

  CDL.Interfaces.RealInput yMax "Maximum allowed output value”;

  CDL.Interfaces.RealInput e "Control error";

  CDL.Interfaces.RealOutput y "Control signal";

  CDL.Continuous.Gain gain(final k=k) "Constant gain";

  CDL.Continuous.Min minValue
    “Outputs the minimum of its inputs";
equation
  connect(yMax, minValue.u1);
  connect(e, gain.u);
  connect(gain.y, minValue.u2);
  connect(minValue.y, y);

  annotation (Documentation(info="<html>
<p>
Block that output <code>y = min(yMax, k*e)</code>,
where
<code>yMax</code> and <code>e</code> are real-valued input
signals and
<code>k</code> is a parameter.
</p>
</html>"));
end CustomPWithLimiter;
```

Graphical annotations omitted.

CDL is used to implement open and proprietary sequences

The standard to be supported by vendors

- **CDL**
- **ASHRAE**
- **G36**

Sequences that come out of ASHRAE projects and can be shared with community.

Custom implementations that are built using the CDL language, and CDL blocks

- **GSA**
- **ARUP**
- **ALC**

GSA preferred sequences, made available through a CDL-complaint implementation.

Design firms can share their own (proprietary) implementation across their offices.

Control vendors can provide their own specialized sequences, either as open-source, or as compiled (proprietary) I/O blocks.

Companies are illustrative

Questions and feedback

1. Block for sunrise/sunset

   Is is sufficient to output the solar angle, or do we need another block that outputs the next time that the sun sets or rises?
   Note that this could be weeks in the future if above the polar circle.

2. What blocks if any are missing or should be removed from CDL?

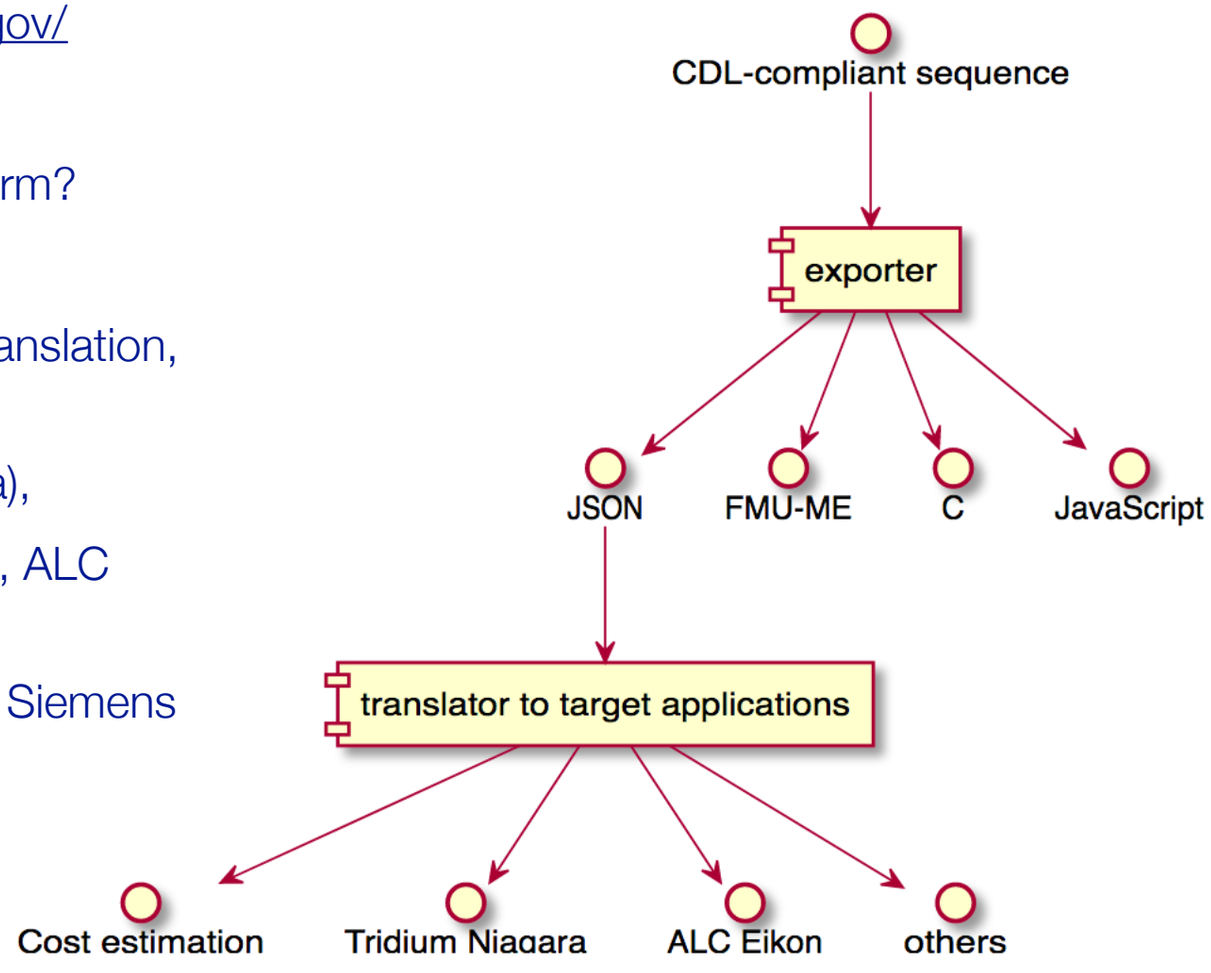
3. Are there any gaps in the CDL specification at http://obc.lbl.gov/specification/cdl.html?

4. Do you see any barrier to translate CDL to your control platform?
   If yes, what are they, how can they be overcome?

5. We need to select *one* target platform for proof of concept translation, such as
   - a free programmable environment (e.g., Tridium Niagara),
   - an environment with block-diagram representation (e.g., ALC Eikon), or
   - an environment with a Process Control Language (e.g., Siemens Apogee).

   This likely needs support from control companies.

6. Any other feedback?



TAG mailing list: https://groups.google.com/forum/#!forum/openbuildingcontrol-tag