A Component Connection Modeling Case Study: Results, and Future Directions

Xudong Yu, Stefanos Manganaris, and Gautam Biswas

Dept. of Computer Science
Vanderbilt University
Nashville, Tennessee 37235

Tel. (615)343–6204

e-mail: xudongy, stefanos, biswas @vuse.vanderbilt.edu

April 16, 1991

Abstract

This paper is a case study in component-connection modeling using CC. We have developed a fairly complex model of a part of the pneumatic system of a DC-10 aircraft for the purpose of behavior generation and analysis. We discuss issues related to modeling and simulation with CC and QSIM, current limitations of CC, and our extentions to overcome them. We show that qualitative component-connection modeling requires global information at the system level in addition to component connection specifications, and that this is the origin for most of the observed limitations.

1 Introduction

A lot of the early research in qualitative reasoning focused on the behavior generation task using envisionment and simulation techniques [12]. The implicit assumption was that appropriate models of physical systems under study were readily available or could be constructed quite easily. However, as the field matured. the emphasis shifted to modeling complex systems and developing real-world applications for design, diagnosis, monitoring, and training. As a result, the focus is now on integrated methods that combine model building and behavior generation.

Two major modeling ontologies are prevalent: the *device-centered* approach [3], and the *process-centered* approach [4]. A third approach, the *constraint-based* approach of QSIM [7] does not directly model a physical situation [8] but performs qualitative simulation on a set of qualitative constraints and an initial state specification provided by the user. Recently there have been two different attempts to develop model-building front-ends to QSIM: (i) QPC [8] creates QSIM QDE's by drawing on a library of model fragments represented as views and processes, and (ii) CC (Component Connection) modeling [5] extends the expressive power and capabilities of the device-oriented approach. The basic units for system building in CC are *primitive components*. Model building is based on the assumption that components only interact via their interface (terminal points). These interactions are represented as *interconnections* between components. Behavior generation involves the derivation of global QDE sets from individual constraint definitions. which are then passed to QSIM for analysis.

Our research focuses on the application of qualitative reasoning methodologies to diagnosis and maintenance training. We require the ability to relate observed faulty behavior of a system to malfunctions of individual components. Therefore, it is important that behaviors of individual components are explicitly defined, and the relation between misbehaving components and overall behavior can be established by qualitative simulation. As part of these efforts, we have built a qualitative model of a part of the *pneumatic system* of a DC-10 aircraft in the CC modeling framework, and derived system behavior using QSIM. In this paper we discuss: (i) how we built the model, and (ii) issues related to modeling and simulation using CC and QSIM. We present current limitations of CC, and our extensions to overcome them. We show that qualitative component-connection modeling requires global information other than just connection specifications and that the lack of this global information is the origin for most of the observed limitations.

2 The Pneumatic System

The schematic of the pneumatic system of a DC-10 aircraft is shown in Fig. 1. For the implementation described in this paper, we focus on the part of the system that regulates temperature and pressure. This part works in cooperation with each one of the three jet engines of the aircraft. At the output, pressure and temperature regulated air flow is available, which is carried through the manifold system to units that constitute the load.

For the purposes of our model, the engine is modeled at a very high level, as two *independent* ideal sources of pressure (in contrast to ideal sources of flow). This is justified by the fact that the power carried by the two pneumatic flows is minimal as compared to the total power delivered by the engine. The pressure delivered by the engine is a function of the throttle position, therefore, in the simulation, the user has explicit control of these two pressures. The pneumatic manifold and the actual loads are lumped together as a sink-type load. The pneumatic controller monitors the operation of the system and controls the operation of the various components. We model almost

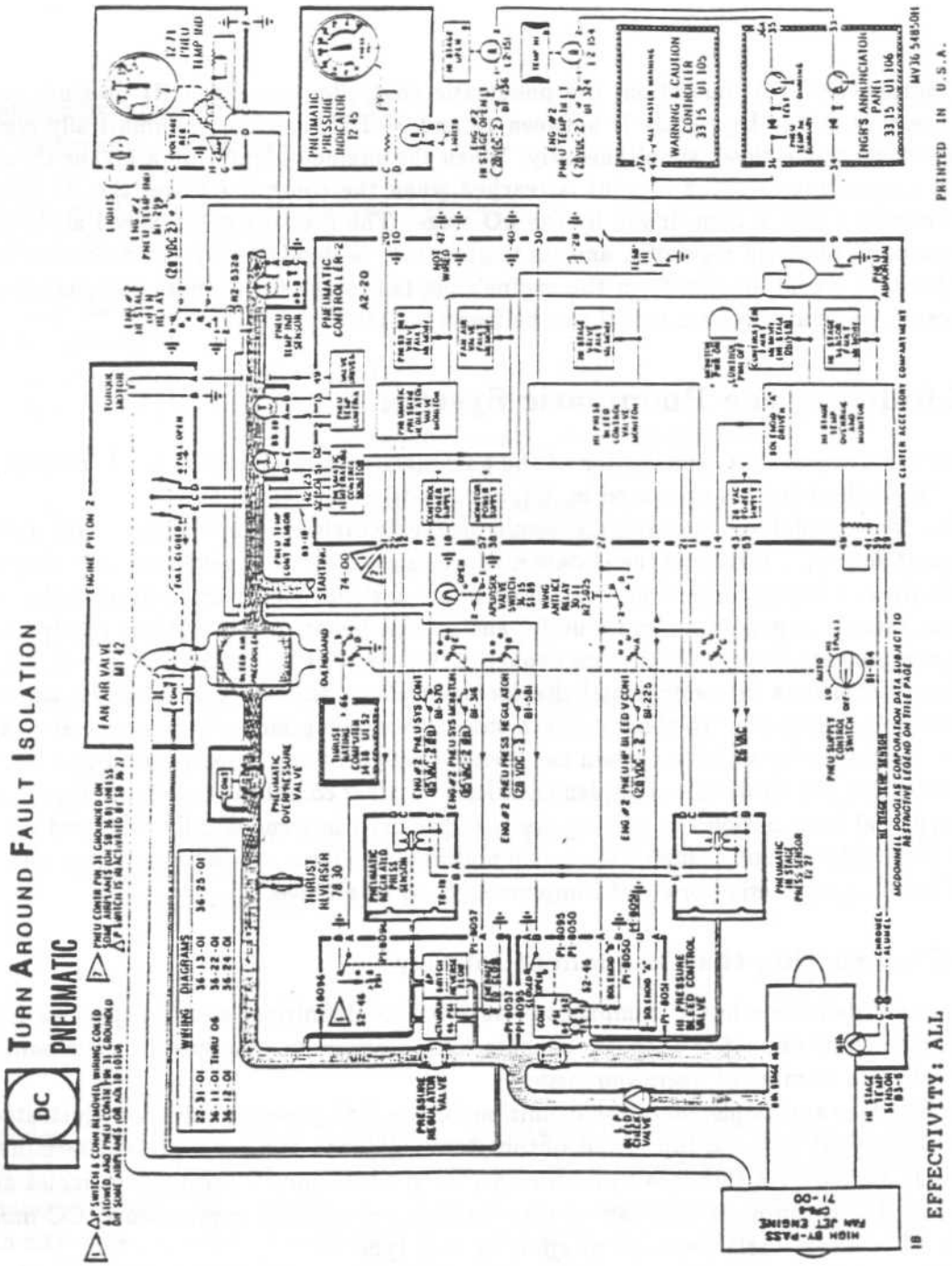

Figure 1: The Pneumatic System of a DC-10 aircraft

all of the functionality of the controller without explicitly modeling the controller in the electrical domain. Also, the various sensors are modeled implicitly. The following components are modeled in detail: the high pressure bleed control valve, the low pressure bleed check valve, the pressure regulator, and the bleed air precooler with the associated fan air valve.

The pressure at *Stage 14* (the HI stage) is always higher than at *Stage 8* (the LO stage). When the engine is idling, pressure at the LO stage is not sufficient for driving the load. Then,

under normal (AUTO) operation, the pneumatic controller opens the HI pressure control valve when pressure at the high stage is between 12 and 84 PSI, and the pneumatically controlled low pressure check valve closes simultaneously. When the engine operates on a higher thrust, pressure at both stages increases and a point is reached when the controller closes the HI pressure valve. The pneumatic load is then driven by the LO stage. The pressure of the bleed air is regulated by the pneumatic pressure regulator, and its temperature is regulated by the precooler system. The precooler uses a cold air flow from the engine's big fan (stage 0). A more complete description of the overall pneumatic system model can be found in [13].

3 Modeling the Pneumatic System

This section discusses implementation of the pneumatic system model in CC [1]. Formal description of the CC methodology is presented in [5], and we do not repeat it here.

The task of model building with CC would *ideally* be divided into two parts: (i) constructing the *component library*, (ii) building the system model as a *composed component* by selecting components from the library and defining interconnections between them. However, to develop the capability of using the model for generating more useful and precise behavior descriptions requires interactions among components to be defined more explicitly. CC handles this by allowing multiple viewpoints and mode definitions in component behavior description, and the ability to dynamically change modes during simulation. To allow interactions between components to be expressed explicitly, we introduce methods by which a system can switch configurations based on user-specified *transition functions*. (See [13] for details of implementation). Further complications that arise because certain kinds of global information are missing are also discussed in some detail. The need for this global system level information complicates a component connection modeling scheme, and makes the task of building a generic library of components hard to achieve.

3.1 Constructing the Component Library

A component definition in CC comprises of an interface definition, and the specification of one or more implementations [5]. Each implementation corresponds to a specific *view point*, which in turn can have a number of *operating modes*.

The interface and implementation definition for the high pressure valve are illustrated in Fig. 2. Note that we split physical input and output terminals into two, to represent terminal variables in different domains (in this case hydraulic and thermodynamic)[2]. Terminal variables are the only parameters of a component that can be directly changed by other components. CC modeling also requires all terminal variables to be of *effort* or *flow* type.

Component implementations can be *primitive* or *composed*. A primitive component implementation defines the variables (terminal or component), and describes behavior as sets of QSIM constraints, with each set corresponding to a specific operating mode of the device. An explicit condition may also be specified for each mode. If the condition is `NIL` this represents a *static* mode definition, otherwise the mode is considered to be *dynamic*. In the dynamic mode, conditions that define the modes are stated explicitly, and this allows the system to determine when a component

[1] The original CC code was obtained from the Qualitative Reasoning Group at the University of Texas at Austin [6]

[2] A more recent version of CC permits now devices to span multiple domains. Other enchancements: *mixed* implementations, support for hierarchical variable names, several new variable options, hierarchical quantity spaces.

```
;;; HI PRESSURE BLEED CONTROL VALVE
(define-component-interface
  HP-Valve
  "Hi Pressure Bleed Control Valve" hydraulic
  (terminals in out in-t out-t))

(define-component-implementation
  primitive HP-Valve
  "Hi Pressure Bleed Control Valve"
  (terminal-variables (in (Qin flow display)
                          (Pin pressure independent display))
                      (out (Qout flow)
                           (Pout pressure display))
                      (in-t (Tin pressure independent))
                      (out-t (Tout pressure))
                      )
  (constraints
   (open NIL
         ((MINUS Qin Qout) (minf inf) (0 0) (inf minf))
         ((= Pin Pout))
         ((= Tin Tout)))
   (closed NIL
           ; The transition function has to specify that Q=0
           ((ZERO-STD Qin))
           ((ZERO-STD Qout)))
  )
)
```

Figure 2: Definition of the High Pressure Valve

change modes. The valve in Fig. 2 is defined to have two static operating modes: *open* and *closed*. In section 3.2.3 we discuss limitations and issues involved in supporting dynamic modes.

A composed component implementation defines a component in terms of other components, which can either be primitive or composed. The pneumatic system model consists of six primitives components. Full component descriptions appear in [13].

3.2 The Composed System

The definition of the composed pneumatic system model in terms of its individual components and their interconnections is presented in Quantity space definitions of individual variables are specified. A list of variables whose initial values are required to define the initial state for QSIM simulation are also specified here. Notice, that this selection of quantity spaces and of a set of variables sufficient for defining a unique initial state for the composed component requires global information.

Fig. 3.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;    THE SYSTEM    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; THE DC-10 PNEUMATIC SYSTEM

(define-component-interface
  Pneumatic-System "The Pneumatic System" hydraulic)

(define-component-implementation
  Composed Pneumatic-System
  "The Pneumatic System"
  (components (Engine CF6-50
                      (landmarks (PH (0 Phi* inf))
                                 (QH (minf 0 inf))
                                 (QL (minf 0 inf))
                                 (PL (0 Plo* plo+ inf))
                                 (AD (0 AD* inf))
                                 (TR (0 TR* TR+ inf))
                                 (NP (minf 0 inf))
                                 (TH (0 Thi* T742 inf))
                                 (TL (0 Tlo* inf))
                                 (TAMB (0 Tamb* inf))
                                 )
                      (initable TR PL TH TL TAMB))
              (LP-Valve LP-Valve
                     (landmarks (Pin (0 Plo* Plo+ inf))
                                (Pout (0 inf))
                                (Pdiff (minf 0 inf))
                                (Qin (minf 0 inf))
                                (Qout (minf 0 inf))
                                (Tin (0 Tlo* inf))
                                (Tout (0 inf))
                                ))
              (HP-Valve HP-Valve
                     (landmarks (Pin (0 Phi* P12 P84 P142 inf))
                                (Pout (0 inf))
                                (Qin (minf 0 inf))
                                (Qout (minf 0 inf))
                                (Tin (0 Thi* T742 inf))
                                (Tout (0 inf))
                                ))
              (Regulator Pressure-regulator
                         (landmarks (Pin (0 Pin* P12 inf))
                                    (Pout (0 inf))
                                    (P (minf 0 inf))
                                    (Qin (minf 0 inf))
                                    (Qout (minf 0 inf))
                                    (PQ (minf 0 inf))
                                    (R (0 inf))
                                    (X (0 XM inf))
                                    (Fs (minf 0 inf))
                                    (dX (minf 0 inf))
                                    (dPout (minf 0 inf))
                                    (Tin (0 Tlo* Thi* inf))
                                 ;  (Tin (0 inf))
                                    (Tout (0 inf)))
                         (initable R))
              (Precooler precooler
                         (landmarks (QHI (minf 0 inf))
                                    (QHO (minf 0 inf))
                                    (PHI (0 P46 PP inf))
                                    (PHO (0 inf))
                                    (THI (0 Tlo* Thi* inf))
                                    (THO (0 T436 inf))
                                    (TCI (0 Tamb* inf))
                                    (TCO (0 inf))
                                    (QCI (0 inf))
                                    (DTH (minf 0 inf))
                                    (DTC (minf 0 inf))
                                    (HFX (minf 0 inf)) ; Positive when from hot to cold
                                    (DT (minf 0 inf))  ; Positive when Hot "hotter" than
cold
                                    (R (0 inf))
                                    (TSET (0 T436 inf))
                                    (ERROR (minf 0 inf)) ; Positive when higer than TSET
                                    (dQCI (minf 0 inf)))
                         (initable R TSET ERROR DT))
              (Load Pneumatic-Load
                    (landmarks (P (0 P46 PP inf))
                               (Q (minf 0 inf))
                               (T (0 T436 inf)))))
  (connections (e-hv (Engine Stage-14) (HP-Valve in))
               (e-hv_t (Engine Stage-14-t) (HP-Valve in-t))
               (e-lv (Engine Stage-8) (LP-Valve in))
               (e-lv_t (Engine Stage-8-t) (LP-Valve in-t))
               (hv-lv-r (HP-Valve out) (LP-Valve out) (Regulator in))
               (hv-lv-r_t (HP-Valve out-t) (LP-Valve out-t) (Regulator in-t))
               (r-p (Regulator out) (Precooler hot-in))
               (r-p_t (Regulator out-t) (Precooler hot-in-t))
               (p-l (Precooler hot-out) (Load in))
               (p-l_t (Precooler hot-out-t) (Load in-t))
               (e-p_t (Engine Stage-1-t) (Precooler cold-in-t))
               )
)
```

Figure 3: Definition of the Pneumatic System of a DC-10 aircraft

Component interactions are defined by interconnections. For example. as shown in Fig. 3, connection *hv-lv-r* connects the three components through their respective terminals: *HP-valve-out*, *LP-valve-out* and *Regulator-in*. Connections imply that the corresponding effort- and flow- variables of these components can interact. More specifically, this interaction is defined by combining terminal variables of these components. To ensure conservation of flow (i.e., *Kirchoff's Current Law (KCL)*), flow type variables (e.g., fluid-flow, electrical current, and heat-flow) are combined using a `minus` QSIM constraint[3]. Variables of the effort type (e.g., pressure, and temperature) are *unified* as described in [5]. Unification substitutes all terminal effort variables of the same domain involved in a connection should be substituted by a common effort variable.

3.2.1 Unification of effort-type variables

An important consideration when unifying effort terminal variables has to do with determining whether the new variable generated is *independent* or *dependent* in the composed system. A variable defined as independent for an individual component may no longer remain independent when it is unified with variables from other components. For example, in the pneumatic system, variable `PHI`, i.e., the input pressure at terminal `hot-in` of the *precooler* is defined as independent for the component. However, when this terminal is connected with the *hot-out* terminal of the pressure regulator, PHI has to be unified with variable `Pout` of the pressure regulator which is not independent. The original CC implementation [5] set the new variable as independent in the generated QDE, if any of the variables to be unified was independent. This forced the modeler to think ahead in terms of specifying which parameters were independent when building the component library. Since this process is dependent on the composed system being considered, the component library can no longer be considered generic. In our approach we try to avoid this think-ahead task, and so the new variable is set to be independent *only* when *all* variables unified are locally independent.

A second significant issue arises when effort variables are unified. Variables often have different *quantity spaces*. For example, the quantity spaces of `PHI` and `Pout` are different. The quantity space associated with the new variable has to be a unification of the individual quantity spaces of the variables being unified. The way quantity spaces are unified significantly impacts the behaviors generated by QSIM. The original version of CC assigned for the new effort, the quantity space of one of the independent variables being unified, or `(minf 0 inf)` if all variables were dependent.

In our approach, the quantity spaces of *all* independent variables have to be considered, and merged into one quantity space that defines the total order of all the landmarks. If partial orders between individual landmarks are available, this will generate multiple quantity spaces, and branching in the model generation phase would have to occur for each different consistent total order. Currently, we automatically merge spaces that coincide, and in other cases, we request help from the user to perform merging during run-time. We do not allow branching on multiple consistent total orderings. As a future task, we will consider the use of quantity-space hierarchies for relating landmarks belonging to different spaces, and the use of the EQUAL primitive instead of unification.

Note again that the issue arises because global information which relates landmarks of quantity spaces belonging to different components is required. An easy way to resolve this problem is to have available the quantitative values associated with each landmark point. Merging of quantity spaces would then become trivial, and no additional global information would be required.

[3] By convention, flow into a component is termed positive.

3.2.2 Propagating Landmarks

A closely related issue to the one presented in the previous subsection has to do with the propagation of landmarks which are "equal". As an example, consider the case of an open valve where input pressure is equal to the output pressure. A similar situation arises in the use of the MINUS to predicate combine terminal flow variables of connected components.

For the case of propagating flow-landmarks, we let the system dynamically generate correspondence values associated with the MINUS constraints after we ensure that related pressure landmarks are propagated properly. In general, since flows cannot be imposed basically by Kirchoff's Laws, we let QSIM handle them after the flat model has been generated by CC.

For propagating effort-landmarks, one can use either MINUS constraints along with established correspondences as in the case of flows described before, or use the EQUAL constraint along with appropriate equivalence introduced into the quantity-space hierarchies. In our modeling work, we have followed the second approach after extending the CC code to allow for the use of quantity space hierarchies.

However, propagating landmarks is not a simple task. In order to propagate a landmark from a component A to another component B, that landmark has to be explicitly included in the quantity spaces or quantity space hierarchies of *all* corresponding parameters of components that lie between the two components. In other words, the quantity space for the parameters of individual components have to be defined in consideration of other components in the system that they may not be directly connected to. There is no defined method for achieving this in CC, so in most instances it has to be done by the system modeler. Again, the modeler has to think ahead and include global information when defining individual components.

For example, when building the component library for the pneumatic system, it was clear that there are certain landmark values that are important to the variables T_{in} and T_{out} of the *precooler*, namely T436, the desired temperature for the bleed air flowing out of the system. The air temperature is determined by the pneumatic source, i.e., the engine. Typically, the temperature of the air originating from the engine is much higher than the desired temperature (T436). In order to ensure that this relationship is established so that appropriate precooler functions are activated, T436 must be included in the quantity space or quantity space hierarchy of the engine output temperatures (TL and TH) as well as temperature parameters of all the components that are on the flow path from engine to the precooler. (All these effort parameters are linked together by unification of terminal parameters of directly connected components, and the *equal* predicate used as a constraint within individual components). If this landmark is missing from any of the quantity spaces (or the quantity space hierarchy) of those components, it would generate multiple initial states for the simulation. For example, given the quantity spaces shown below, the qspace hierarchy definition shown in Fig. 4, and the initial value for `TL` of engine as `Tlo*` (`TH` is not important at this time, because the HP-Valve is not open yet), more than one initial state is generated by QSIM. Two of the states are shown in Fig. 5.

Engine	`TL`: (0 T436 Tlo* inf)	`TH`: (0 Thi* T742 inf)
LP-Valve	`Tin`: (0 T436 Tlo* inf)	`Tout`: (0 inf)
HP-Valve	`Tin`: (0 Thi* T742 inf)	`Tout`: (0 inf)
Regulator	`Tin`: (0 inf)	`Tout`: (0 inf)
Precooler	`THI`: (0 Tlo* Thi* inf)	`THO`: (0 T436 inf)

```
(defconstant pneumatic-system-other
    '(other
       (qspace-hierarchy
          ((PNEUMATIC-SYSTEM_COMPOSED.EFFORTHV-LV-R_T
          PNEUMATIC-SYSTEM_COMPOSED.EFFORTE-LV_T
          PNEUMATIC-SYSTEM_COMPOSED.EFFORTE-HV_T
          PNEUMATIC-SYSTEM_COMPOSED.EFFORTR-P_T) ->
          (*seq 0 T436 TLO* THI* inf))
    )))
```

Figure 4: The qspace Hierarchy

Note that variables in the qspace hierarchy in Fig. 4 represent names of unified variables at different connection points.

As we can see from the figure, even though the temperature at the engine was known to be `Tlo*` (higher than T436) and it was not changed along the flow path, QSIM did not know what the temperature at the input terminal of the precooler was. QSIM generated a landmark for `THI` of the precooler, which can be anything in its quantity space ($<$, $=$, $>$ any of the other landmarks in its quantity space). A possible solution to this problem besides including the landmark (e.g., `Tlo*`) in all quantity spaces during model building is to set up mechanisms in CC so that whenever a quantity space hierarchy is established for unification, these landmarks are automatically added to the quantity spaces of the parameters involved.

3.2.3 System Configurations and Transitions

Since individual components can operate in multiple modes, the overall system can dynamically assume a number of different *configurations*. Theoretically, if the composed system contains n components each with m modes, m^n different configurations should be attainable by the system. For example, the pneumatic system could operate in four configurations depending on the HP- and LP- valve modes. However, only two of these are attained during normal operation because both valves cannot be opened or closed at the same time when the engine operates. (The other two configurations may be useful to describe faulty situations though.) A configuration definition specifies the implementation and operating mode of each component comprising the system. For example, in Fig. 6, the configuration *HP-C-LP-O* defines an configuration of the pneumatic system with the HP-Valve closed and the LP-Valve opened.

Whenever a component changes its operating mode, the system must necessarily switch from one configuration to another. An implementation requires global information here too. First, information about the current configuration, in particular about the modes of the other components, has to be used in order to determine the resulting configuration. Second, global information may be required to determine a consistent system state in the new configuration after the transition.

Automating these two tasks without explicitly modeling the transition itself at a lower level is impossible. A change in the operating mode of one component may be abrupt (discontinuous) and result in a simultaneous change of the operating mode of other components before a consistent state can be reached. In the pneumatic system, closing the HP- valve results in a simultaneous opening of the LP- valve. There is no consistent state in any intermediate configuration of the

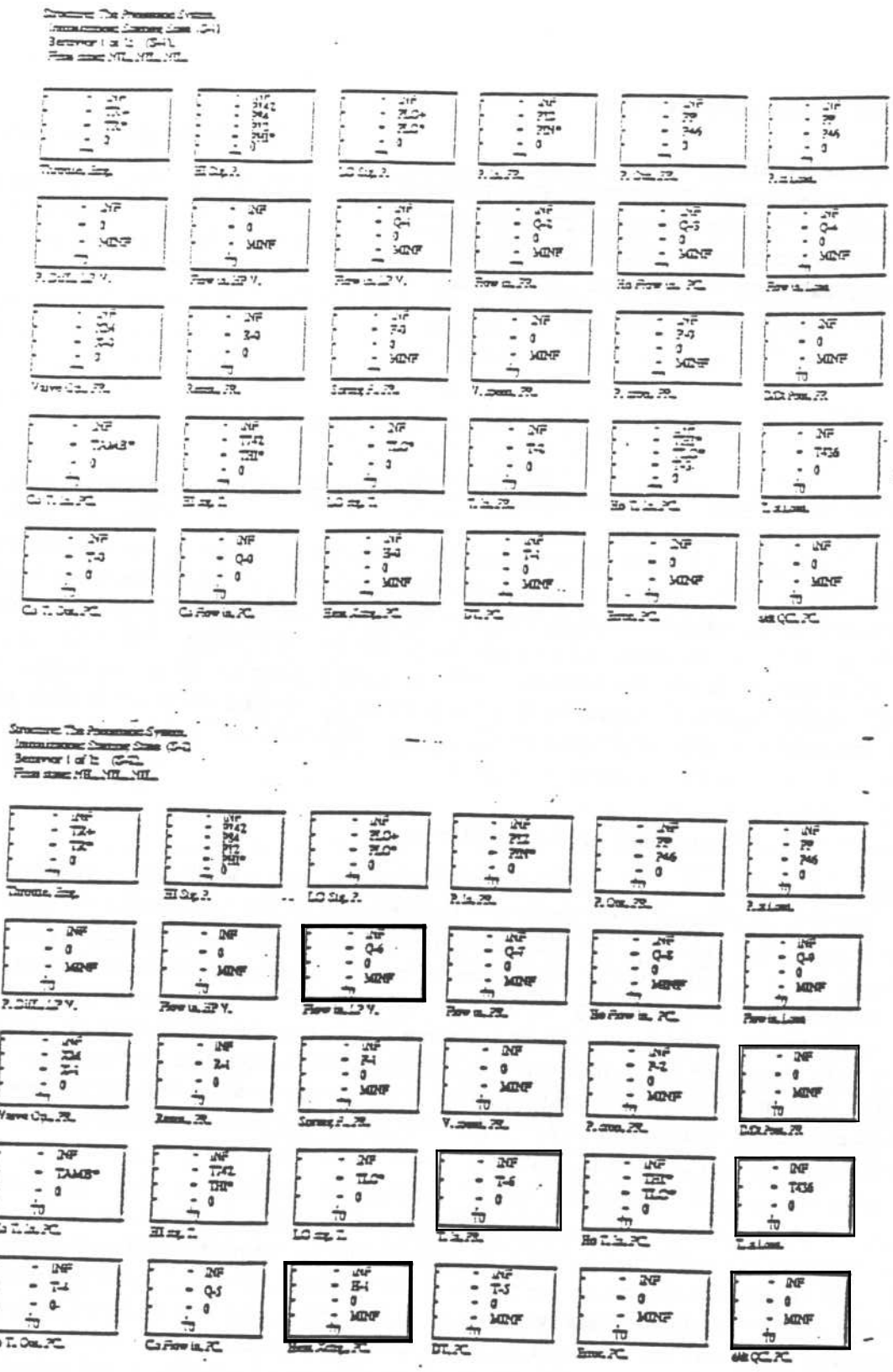

Figure 5: Two of the multiple initial states

```
(define-configuration HP-C-LP-O
  Pneumatic-System "HP Valve CLOSED and LP Valve OPEN"
  (HP-Valve (impl primitive) (mode closed))
  (LP-Valve (impl primitive) (mode open))
  )

(defconstant HP-C-LP-O-transitions
  '(transitions
    (Open-HP-VALVE-T Open-HP-VALVE)
    (Close-LP-VALVE-T Close-LP-VALVE)
    ))
```

Figure 6: A Configuration of the Pneumatic System and Associated Transitions

current model. It is also difficult to derive a consistent system state after a transition has occurred. A transition, in general, causes an immediate and possibly discontinuous, changes in a number of variables. Even if these changes are specified, it is necessary to determine: (i) which subset of variables do not change values in the transition, and (ii) the remaining variables whose values are computed by QSIM to create a consistent initial state. This is a hard task to achieve especially since one want to avoid an inconsistent initial state in the new configuration, or excessive spurious branching because parameter values were not specified.

Recognizing the fact that local information is not enough for implementing dynamic operating modes, support from CC of QSIM's transition mechanism was not provided in the original version. In our extensions, we simply provide the information required in the form of:

- A transitions clause associated with each configuration that specifies what transitions can possibly occur while the system is in the particular configuration,
- Transition test functions that detect when a transition is activated.
- A QDE transition table, that specifies for each one of the possible current configurations and for each transition the resulting configuration (Fig 7), and
- Transition functions, that are responsible for generating the QDE for the new configuration (if not already generated) and the resulting system's state.

We use the mechanism of QSIM's transitions to do the rest. Notice that we provide support for multiple QDE's, and that the names of variables are not dependent on a particular configuration.

4 Behavior Generation by Qualitative Simulation

Behavior generation via simulation requires definition of the operating conditions of the system, that is, the settings of controls, initial values of initable internal parameters, values of external parameters, etc. QSIM derives from that description using the model generated by CC all possible behaviors of the system. A number of simulation runs have been conducted. Here, we discuss one of them in detail. A couple of behaviors generated from this simulation are shown in Fig. 8.

```
(defconstant TRANSITION-TABLE
  '(
     (HP-C-LP-C_PNEUMATIC-SYSTEM_COMPOSED ((O-HP-VALVE HP-O-LP-C)
                                           (C-HP-VALVE NIL)
                                           (O-LP-VALVE HP-C-LP-O)
                                           (C-LP-VALVE NIL)))
     (HP-O-LP-C_PNEUMATIC-SYSTEM_COMPOSED ((O-HP-VALVE NIL)
                                           (C-HP-VALVE HP-C-LP-C)
                                           (O-LP-VALVE HP-O-LP-O)
                                           (C-LP-VALVE NIL)))
     (HP-C-LP-O_PNEUMATIC-SYSTEM_COMPOSED  ((O-HP-VALVE HP-O-LP-C)
                                            (C-HP-VALVE NIL)
                                            (O-LP-VALVE NIL)
                                            (C-LP-VALVE HP-C-LP-C)))
     (HP-O-LP-O_PNEUMATIC-SYSTEM_COMPOSED  ((O-HP-VALVE NIL)
                                            (C-HP-VALVE HP-C-LP-O)
                                            (O-LP-VALVE NIL)
                                            (C-LP-VALVE HP-O-LP-C)))
     ))
```

Figure 7: Transition Table Definition

In this simulation, the pneumatic supply control switch (PSCS) is set to AUTO, wing anti-ice is set to ON, and control and monitor power is supplied to the controller. The simulation starts from an equilibrium point with the engine idling. The HI Stage pressure is below 12 PSI, and thus the HI pressure valve is closed. At time T1, the throttle is increased and the behavior generation process was initiated. When HI Stage pressure exceeded 12 PSI (landmark P12), a transition occurred. The HP-valve opened and the LP-valve closed. The transition manifested itself as a discontinuous change in temperature and pressure at the input to the pressure regulator. Pressure at the output of the pressure regulator increased but then stabilized, in the sense that it remained in the same qualitative interval. Pressure at the load stabilized too. The temperature of the air goes through a similar cycle. When HI Stage pressure reaches 142 PSI (P142), the controls cause the HP valve to close, the LP valve opens and the pressure at input of the pressure regulator suddenly drops back to 12 PSI (we assume a negligible increase in LO Stage pressure). The two behaviors in Fig. 8 indicate qualitatively different responses to the same throttle position change. A number of other simulation runs conducted are discussed in [13].

5 Discussion

This case study demonstrates that the CC approach can be successfully used to model complex real-world systems for behavior generation and analysis, provided that the required global information indicated in the paper is also supplied. Limitations in the expressiveness of the CC language and their effect on modeling and behavior generation were demonstrated. One way to address some of the problems would be to associate quantitative values with landmark points. As discussed before,

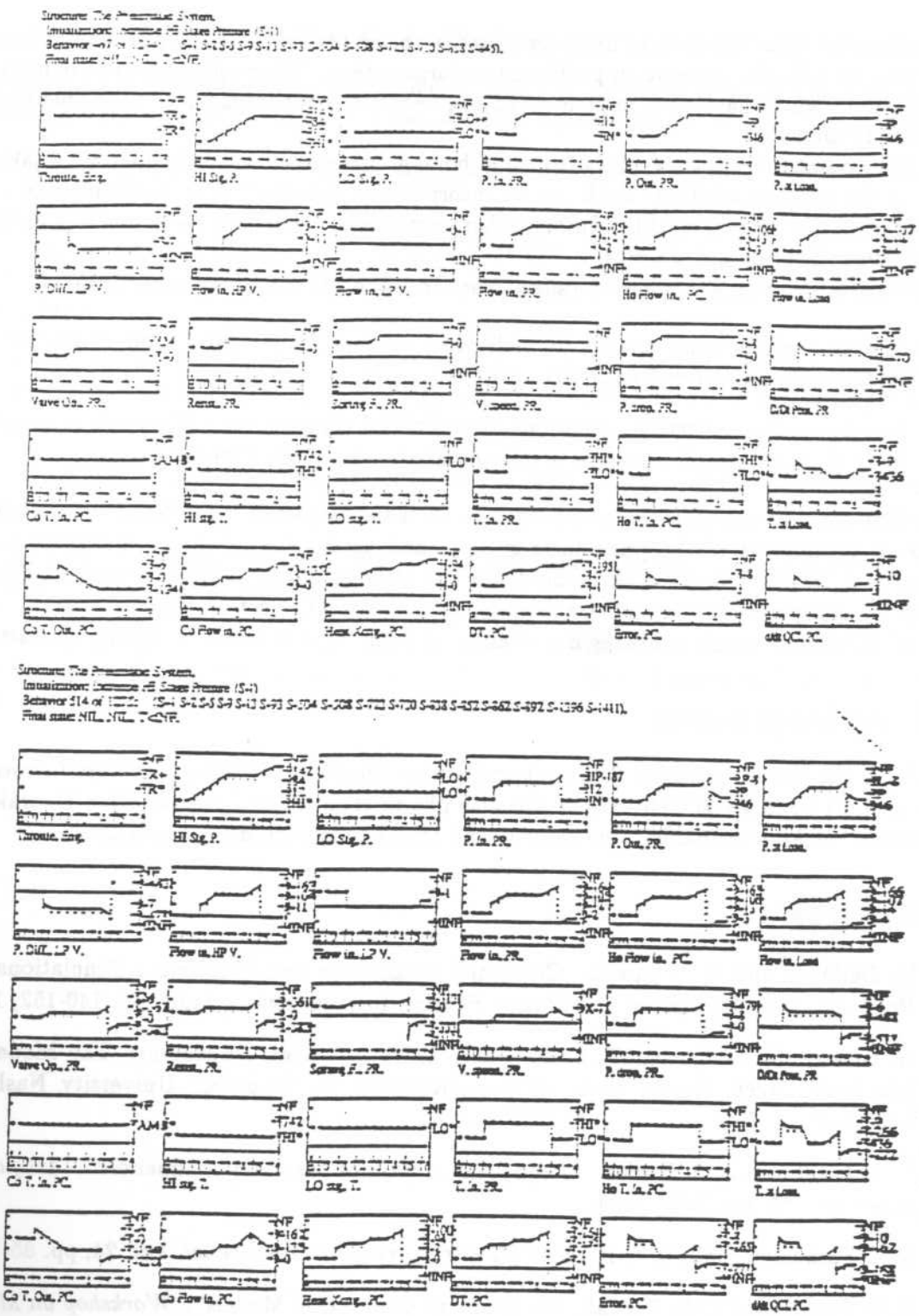

Figure 8: Pneumatic System Responses to Increase in HI Stage Pressure

this would trivialize the task of unifying quantity spaces. In fact a Q3-like scheme [1] may address the multiple behavior generation problem to a large extent. Other approaches have been discussed in the paper, which have been implemented and tested in modeling a part of the pneumatic system of a DC-10 aircraft.

CC borrows concepts and ideas from the bond graph modeling methodology [9]. We feel, that more of the bond-graph framework can be incorporated into qualitative modeling and simulation. Part of our work focuses on this. Among other advantages, we believe there can be significant gains in causal explanation capabilities.

We are looking at a number of issues to enhance the CC modeling framework:

- How to combine bond graphs for individual elements to produce bond graphs for composed systems (modular bond-graphs).
- How to combine constraint relations for individual components with causal relations obtained from the bond graph for more precise qualitative behavior generation.

As a first step in establish this methodology, we have implemented a bond graph model of the precooler component of the pneumatic system, and used it to develop a model-based diagnosis scheme [14]. Other related work involves the use of bond graph and perturbation analysis mechanisms in deriving analysis and explanation generation scheme for training system. A detailed discussion on bond graph modeling and their application to qualitative reasoning appears in [10,11].

Acknowledgements

We thank the Air Operations, Information Services Division at Federal Express Corporation, for their support in this research. We would also like to thank David Franke, Dan Dvorak, and Ben Kuipers for making available to us the CC and QSIM code with documentation.

References

[1] D. Berleant and B. Kuipers. Combined Qualitative and Numerical Simulations with Q3. *Proc. 4th. Intl. Wkshp. on Qualitative Physics*, Lugano, Switzerland, pp. 140-152, 1990.

[2] G. Biswas, N. McQueen, T. Crews, and X. Yu. "Design Document: The Model Building Interface", *Tech. Report*, Dept. of Computer Science, Vanderbilt University, Nashville, TN, Dec. 1990.

[3] J. de Kleer and J. S. Brown. "A qualitative physics based on confluences", *Artificial Intelligence*, vol. 24, pp. 7-83, 1984.

[4] K.D. Forbus. "Qualitative Process Theory", *Artificial Intelligence*, vol. 24, pp. 85-168, 1984.

[5] D.W. Franke and D.L. Dvorak. "Component Connection Models", *Workshop on Model-Based Reasoning*, IJCAI-89, Detroit, MI, pp. 97-101, 1989.

[6] D.W. Franke and D.L. Dvorak. "CC: Component Connection Models for Qualitative Simulation, A User's Guide", *Tech. Report*, University of Texas at Austin, Austin, Texas, October 1989.

[7] B. Kuipers. "Qualitative Simulation", *Artificial Intelligence*, vol. 29, pp. 289-388, 1986.

[8] J. Crawford, A. Farquhar, and B. Kuipers. "QPC: A Compiler from Physical Models into Qualitative Differential Equations", *Proc. AAAI90*, pp. 365-372, 1990.

[9] R.C. Rosenberg, and D.C. Karnopp. Introduction to Physical System Dynamics, McGraw-Hill, 1983.

[10] S. Manganaris and G. Biswas. "Modeling and Qualitative Reasoning about Perturbations in Mechanics Problem Solving", *Tech. Report*, Computer Science Dept., Vanderbilt University, April, 1990.

[11] S. Manganaris and G. Biswas. "ITS for Maintenance Training at Federal Express Corporation: The Mechanism for Explanations", *Tech. Report*, Department of Computer Science, Vanderbilt University, Dec. 1990.

[12] D.S. Weld and .J. de Kleer, eds. *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufmann, San Mateo, CA, 1989.

[13] X. Yu, S. Manganaris, and G. Biswas. "The Pneumatic System", *Tech. Report*, Federal Express Corporation, Memphis, TN, 1990.

[14] X. Yu and G. Biswas. "A Candidate Generation Method for Model-Based Diagnosis of Continuous-valued Systems", in review, *1991 Model-Based Reasoning Workshop (AAAI)*, Anaheim, CA, July 1991.

APPENDIX

```lisp
;;; @(#) pneumatic-components.lisp 1.20@(#)

;;; -*- Mode: LISP; Syntax: Common-lisp; Package: QSIM; Base: 10 -*-

;;; (The "engineers" database)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  COMPONENT LIBRARY  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;; CF6-50 JET ENGINE
(define-component-interface
  CF6-50
  "Engine, as far as the pneumatic system is concerned" hydraulic
  (terminals Stage-8 Stage-14))

(define-component-implementation
  primitive CF6-50
  "Engine"
  (terminal-variables (Stage-14 (PH pressure display)
                                (QH flow))
                      (Stage-14-t (TH pressure independent display))
                      (Stage-8 (PL pressure independent display)
                               (QL flow))
                      (Stage-8-t (TL pressure independent display))
                      (Stage-1-t (TAMB pressure independent display)))
  (component-variables (TR amount independent display)  ; Throttle position
                       (AD amount)       ; Inertia of engine (spring-model)
                       (NP amount)       ; Net Power
  )
  (constraints
   ((M+ PH AD) (PHI* AD*) (0 0) (inf inf))
   ((ADD NP AD TR) (0 AD* TR*))
   ((d/dt PH NP))
   )
  )


;;; HI PRESSURE BLEED CONTROL VALVE
(define-component-interface
  HP-Valve
  "Hi Pressure Bleed Control Valve" hydraulic
  (terminals in out in-t out-t))

(define-component-implementation
  primitive HP-Valve
  "Hi Pressure Bleed Control Valve"
  (terminal-variables (in (Qin flow display)
                          (Pin pressure independent display))
                      (out (Qout flow)
                           (Pout pressure display))
                      (in-t (Tin pressure independent))
                      (out-t (Tout pressure))
                      )
  (constraints
   (open NIL
         ((MINUS Qin Qout) (minf inf) (0 0) (inf minf))
         ((= Pin Pout))
         ((= Tin Tout)))
   (closed NIL
           ; The transition function has to specify that Q=0
           ((ZERO-STD Qin))
           ((ZERO-STD Qout)))
   )
  )


;;; LP BLEED CHECK VALVE
(define-component-interface
  LP-Valve
  "Low Pressure Bleed Check Valve" hydraulic
  (terminals in out in-t out-t))

(define-component-implementation
  primitive LP-Valve
  "Low Pressure Bleed Check Valve"
  (terminal-variables (in (Qin flow display)
                          (Pin pressure independent display))
                      (out (Qout flow)
                           (Pout pressure display))
                      (in-t (Tin pressure independent))
                      (out-t (Tout pressure)))
  (component-variables (Pdiff pressure))
  (constraints
   (open NIL
         ((ADD Pdiff Pout Pin) (0 0 0))
         ((ZERO-STD Pdiff))
         ((MINUS Qin Qout) (minf inf) (0 0) (inf minf))
         ((= Pin Pout))
         ((= Tin Tout)))
   (closed NIL
           ; The transition function has to specify that Q=0
           ((ADD Pdiff Pout Pin) (0 0 0))
           ((ZERO-STD Qin))
           ((ZERO-STD Qout)))
   )
  )


;;; PRESSURE REGULATOR VALVE
(define-component-interface
  Pressure-regulator
  "Pressure regulator valve" hydraulic
  (terminals in out in-t out-t))

(define-component-implementation
  primitive Pressure-regulator
  "Pressure-regulator"
  (terminal-variables (in (Qin flow display)
                          (Pin pressure independent display))
                      (out (Qout flow )
                           (Pout pressure display))
                      (in-t (Tin pressure independent))
                      (out-t (Tout pressure)))
  (component-variables (P pressure display)
                       (PQ pressure display)
                       (R amount display)
                       (X amount display)
                       (Fs amount display)
                       (dx amount display)
                       (dPout amount display))
  (constraints ((MINUS Qin Qout) (minf inf) (0 0) (inf minf))
               ((ADD P Pout Pin))
               ((MULT R Qin P))
```

```
              ((M+ Qin PQ) (0 0) (inf inf) (minf minf))
              ((ADD Pout dPout PQ))
              ((d/dt Pout dPout))
              ((M+ X R) (0 0) (XM inf))
              ((M+ X Fs))
              ((ADD Fs dX Pout))
              ((d/dt X dx))
              ((= Tin Tout)))
)


;;; BLEED AIR PRECOOLER & FAN AIR VALVE (including negative feedback controler)
(define-component-interface
  Precooler
  "Bleed Air Precooler and Fan Air Valve"
  (terminals hot-in hot-out cold-in
             hot-in-t hot-out-t cold-in-t))

(define-component-implementation
  primitive Precooler
  "Precooler and feedback control modeled in QSIM primitives"
  (terminal-variables (hot-in (QHI flow display)
                              (PHI pressure independent display))
                      (hot-in-t (THI pressure independent display))
                      (hot-out (QHO flow)
                               (PHO pressure display))
                      (hot-out-t (THO pressure display))
                      (cold-in-t (TCI pressure independent)))
  (component-variables (TCO pressure display)
                       (DTH amount display)
                       (DTC amount display)
                       (HFX amount display)
                       (DT amount display)
                       (R amount independent)
                       (TSET amount independent)
                       (ERROR amount display)
                       (QCI flow display)
                       (dQCI amount display))
  ; history: QCI ERROR DT
  (constraints ((MINUS QHI QHO) (0 0) (inf minf) (minf inf))
               ((= PHI PHO))
               ;((UNREACHABLE THI (VALUES 0 INF)))
               ;((UNREACHABLE THO (VALUES 0 INF)))
               ;((UNREACHABLE TCI (VALUES 0 INF)))
               ;((UNREACHABLE TCO (VALUES 0 INF)))
               ((ADD DTH THO THI))
               ((ADD DTC TCI TCO))
               ((MULT DTH QHI HFX))
               ((MULT DTC QCI HFX))
               ((ADD DT TCO THO))
               ((MULT R DT HFX))
               ((ADD ERROR TSET THO)     (0 T436 T436))
               ((M+ ERROR dQCI) (0 0))
               ((D/DT QCI dQCI)))
  )


;;; DC-10 PNEUMATIC LOAD
(define-component-interface
  Pneumatic-Load
  "Air Conditioning Packs, Anti Ice, Engine Start, etc."
  (terminals in in-t))

(define-component-implementation
  primitive Pneumatic-Load
  "Pneumatic Load modeled as a sink in QSIM primitives"
  (terminal-variables (in (Q flow display)
                          (P pressure independent display))
                      (in-t (T pressure independent display))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   THE SYSTEM    ;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; THE DC-10 PNEUMATIC SYSTEM

(define-component-interface
  Pneumatic-System "The Pneumatic System" hydraulic)

(define-component-implementation
  Composed Pneumatic-System
  "The Pneumatic System"
  (components (Engine CF6-50
                      (landmarks (PH (0 Phi* inf))
                                 (QH (minf 0 inf))
                                 (QL (minf 0 inf))
                                 (PL (0 Plo* plo+ inf))
                                 (AD (0 AD* inf))
                                 (TR (0 TR* TR+ inf))
                                 (NP (minf 0 inf))
                                 (TH (0 Thi* T742 inf))
                                 (TL (0 Tlo* inf))
                                 (TAMB (0 Tamb* inf))
                                 )
                      (initable TR PL TH TL TAMB))
              (LP-Valve LP-Valve
                     (landmarks (Pin (0 Plo* Plo+ inf))
                                (Pout (0 inf))
                                (Pdiff (minf 0 inf))
                                (Qin (minf 0 inf))
                                (Qout (minf 0 inf))
                                (Tin (0 Tlo* inf))
                                (Tout (0 inf))
                                ))
              (HP-Valve HP-Valve
                     (landmarks (Pin (0 Phi* P12 P84 P142 inf))
                                (Pout (0 inf))
                                (Qin (minf 0 inf))
                                (Qout (minf 0 inf))
                                (Tin (0 Thi* T742 inf))
                                (Tout (0 inf))
                                ))
              (Regulator Pressure-regulator
                         (landmarks (Pin (0 Pin* P12 inf))
                                    (Pout (0 inf))
                                    (P (minf 0 inf))
                                    (Qin (minf 0 inf))
                                    (Qout (minf 0 inf))
                                    (PQ (minf 0 inf))
                                    (R (0 inf))
                                    (X (0 XM inf))
                                    (Fs (minf 0 inf))
                                    (dX (minf 0 inf))
                                    (dPout (minf 0 inf))
                                    (Tin (0 Tlo* Thi* inf))
                                 ;  (Tin (0 inf))
                                    (Tout (0 inf)))
                         (initable R))
              (Precooler precooler
                                    (PHO (0 inf))
                                    (THI (0 Tlo* Thi* inf))
                                    (THO (0 T436 inf))
                                    (TCI (0 Tamb* inf))
                                    (TCO (0 inf))
                                    (QCI (0 inf))
                                    (DTH (minf 0 inf))
                                    (DTC (minf 0 inf))
                                    (HFX (minf 0 inf)) ; Positive when from hot to cold
                                    (DT (minf 0 inf))  ; Positive when Hot "hotter" than
cold
                                    (R (0 inf))
                                    (TSET (0 T436 inf))
                                    (ERROR (minf 0 inf)) ; Positive when higer than TSET
                                    (dQCI (minf 0 inf)))
                         (initable R TSET ERROR DT))
              (Load Pneumatic-Load
                    (landmarks (P (0 P46 PP inf))
                               (Q (minf 0 inf))
                               (T (0 T436 inf)))))
  (connections (e-hv (Engine Stage-14) (HP-Valve in))
               (e-hv_t (Engine Stage-14-t) (HP-Valve in-t))
               (e-lv (Engine Stage-8) (LP-Valve in))
               (e-lv_t (Engine Stage-8-t) (LP-Valve in-t))
               (hv-lv-r (HP-Valve out) (LP-Valve out) (Regulator in))
               (hv-lv-r_t (HP-Valve out-t) (LP-Valve out-t) (Regulator in-t))
               (r-p (Regulator out) (Precooler hot-in))
               (r-p_t (Regulator out-t) (Precooler hot-in-t))
               (p-l (Precooler hot-out) (Load in))
               (p-l_t (Precooler hot-out-t) (Load in-t))
               (e-p_t (Engine Stage-1-t) (Precooler cold-in-t))
               )
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;  TRANSITION and TRANSITION-TEST FUNCTIONS  ;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


Test functions:  Either a function which takes as argument the current state
and returns two truth values (pp. 2, transitions.lisp) OR a list of the form
(<variable name> (<landmark> <qdir>)).   Example:

(defconstant Open-LP-valve-T '(PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF (0 inc)))
(don't forget the comma in "deconstant transitions"!)

;; USEFUL THINGS

*zero-lmark*                          ; wild zero
*inf-lmark*                           ; wild inf

; qmag qv, qdir qv, qval vname s      ; (internals)
; qvalue= (vname s)                   ; (civilized)
; qspace vname s                      ; (internals, from state)
; variable--qspace v                  ; (internals, from var) (double minus indicates
changing in sim!)
; lmark-name l                        ; (to civilize wild things)
; qmag-order qmag lmark qspace        ; (internals, returns -/0/+/NIL)
; landmark-le a b qspace              ; (civilized? -- member member eql) NO!
; landmark-lt a b qspace              ; (civilized? -- member member eql) NO!
; convert-user-qmag uqmag qspace      ; (wild zap.  Qspace is wild)
;
;;;


;;; Some useful comments from experience:

; In defining a state after a transition one has to specify for each parameter
; if:
;       - The current qmag of the parameter is to be propagated to the new state.
;       (This is the default)
;       - The new value is to be determined by QSIM (propagation & constraint
;       filtering) (potential branching)
;       - A new value is to be used, as specified by the transition.
;
; In most cases DON'T specify the new flows.  They cannot be determined localy
; so let QSIM do it.


;;; LP VALVE ;;;

(defun CLose-LP-valve-T (state)
  ; Smooth and abrupt transitions are sensed...
  (cond
   ((and (eql (qvalue= (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF state)) 0)
         (eql (qdir (qval 'PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF state)) 'DEC))
    (values T NIL))
   ((eql (qmag-order (qmag (qval 'PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF state))
                     *zero-lmark*
                     (qspace 'PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF state))
         '-)
    (values T NIL))
   ))

(defun Close-LP-valve (state)
  ; needs work
  ;(format *qsim-report* "~&~%Closing LP Valve...")
  (create-transition-state :from-state state
                           :to-qde (build-qde (new-configuration state 'c-lp-valve))
                           :assert '(
                                (PNEUMATIC-SYSTEM_COMPOSED_ENGINE.QL@E-LV (0 std))
                                (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.QIN@E-LV (0 std))
                                (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.QOUT@HV-LV-R (0 std))
                                )
                           :inherit-qmag :rest
                           :inherit-qdir NIL
                           :text "Close LO Pressure Valve"))


(defun Open-LP-valve-T (state)
  ; Smooth and abrupt transitions are sensed...
  (cond
   ((and (eql (qvalue= (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF state)) 0)
         (eql (qdir (qval 'PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF state)) 'INC))
    (values T NIL))
   ((eql (qmag-order (qmag (qval 'PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF state))
                     *zero-lmark*
                     (qspace 'PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF state))
         '+)
    (values T NIL))
   ))

(defun Open-LP-valve (state)
  ;(format *qsim-report* "~&~%Opening LP valve...")
  (create-transition-state :from-state state
                           :to-qde (build-qde (new-configuration state 'o-lp-valve)
                                              :QDE-clauses '(
                                                             ,pneumatic-system-print-name
s
                                                             ,(new-transitions state 'o-l
p-valve)
                                                             ,pneumatic-system-other
                                                             ))
                           :assert '(
                                (PNEUMATIC-SYSTEM_COMPOSED_ENGINE.QL@E-LV (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.QIN@E-LV (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.QOUT@HV-LV-R (nil nil
))
                                (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF (0 nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.P (nil nil))
                                ;(PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.QIN@HV-LV-R (nil ni
l))
                                ;(PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.QOUT@R-P (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.PQ (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.DPOUT (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.X ((0 XM) nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.Fs (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.R (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.QHI@R-P (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.QHO@P-L (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.ERROR (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.TCO (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.DTC (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.DTH (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.HFX (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.dQCI ((minf inf) nil
))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.DT (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_LOAD.Q@P-L (nil nil))
```

```
                        (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@HV-LV-R (Pin* nil)) ;
pin* helps
                        (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-LV (PLO* std))
                        (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@R-P (PP nil))
                        (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@P-L (PP nil))
                        ;(PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-LV_T (TLO* std))
                        (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@HV-LV-R_T (nil nil))
                        (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@R-P_T (nil nil))
                        (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@P-L_T (nil nil))
                        )
                    :inherit-qmag :rest
                    :inherit-qdir NIL
                    :text "Open LO Pressure Valve"))


;;; HP VALVE ;;;

(defun CLose-HP-valve-T (state)
  ; Only smooth transitions are sensed.
  (cond
   ((and (eql (qvalue= (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'P12)
         (eql (qdir (qval 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'DEC))
    (values T NIL))
   ((and (eql (qvalue= (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'P84)
         (eql (qdir (qval 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'INC)
         (solenoid-A-energized))
    (values T NIL))
   ((and (eql (qvalue= (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'P142)
         (eql (qdir (qval 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'INC)
         (solenoid-B-energized))
    (values T NIL))
   ((and (eql (qde-name (state-qde state)) 'Hp-o-lp-c_pneumatic-system_composed)
         (eql (qmag-order (qmag (qval 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T state))
                          (convert-user-qmag 'T742
                                             (qspace 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E
-HV_T state))
                          (qspace 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T state))
              '+))
    (values T NIL))
   ))


(defun Close-HP-valve (state)
  ;(format *qsim-report* "~&~%Close HP valve...")
  (create-transition-state :from-state state
                           :to-qde (build-qde (new-configuration state 'c-hp-valve)
                                              :QDE-clauses '(
                                                             ,pneumatic-system-print-nam
es
                                                             ,(new-transitions state 'c-
hp-valve)
                                                             ,pneumatic-system-other
                                                             ))
                           :assert '(
                                (PNEUMATIC-SYSTEM_COMPOSED_ENGINE.QH@E-HV (0 std))
                                (PNEUMATIC-SYSTEM_COMPOSED_HP-VALVE.QIN@E-HV (0 std))
                                (PNEUMATIC-SYSTEM_COMPOSED_HP-VALVE.QOUT@HV-LV-R (0 std)
)
                                (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.QIN@HV-LV-R (nil ni
l))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.QOUT@R-P (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.P (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.PQ (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.DPOUT (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.X (0 nil)) ; Close c
ompletely momentarily, (inertia)
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.Fs (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.R (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.QHI@R-P (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.QHO@P-L (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.ERROR (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.TCO (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.DTC (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.DTH (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.HFX (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.dQCI ((minf inf) nil
))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.DT (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_LOAD.Q@P-L (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@HV-LV-R
                                        (,(new-effort@hv-lv-r state 'c-hp-valve) nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@R-P (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@P-L (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@HV-LV-R_T
                                        (,(new-effort@hv-lv-r_t state 'c-hp-valve) nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@R-P_T (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@P-L_T (nil nil))
                                )
                    :inherit-qmag :rest
                    :inherit-qdir NIL
                    :text "Close HI Pressure Valve"))

(defun new-effort@hv-lv-r (state transition)
  ; 0 if transition is c-hp-valve and lp is closed,  effort@e-lv if lp is open (??)
  (cond
   ((and (eql transition 'c-hp-valve) (eql (qde-name (state-qde state)) 'Hp-o-lp-c_pneuma
tic-system_composed))
    0)
   (T
    (qvalue= (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-LV state)))))

(defun new-effort@hv-lv-r_t (state transition)
  ; 0 if transition is c-hp-valve and lp is closed,  effort@e-lv_t if lp is open (??)
  ; if o-hp-valve then effort@e-hp_t.
  (cond
   ((eql transition 'o-hp-valve)
    (qvalue= (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T state)))
   ((and (eql transition 'c-hp-valve) (eql (qde-name (state-qde state)) 'Hp-o-lp-c_pneuma
tic-system_composed))
    0)
   (T
    (qvalue= (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-LV_T state)))))


(defun Open-HP-valve-T (state)
  ; Only smooth transitions are sensed...
  (cond
   ((and (eql (qvalue= (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'P12)
         (eql (qdir (qval 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'INC)
         (eql (qmag-order (qmag (qval 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T state))
                          (convert-user-qmag 'T742
                                             (qspace 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-
HV_T
                                                     state))
                          (qspace 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T state))
              '-))
```

```
    (values T NIL))
   ((and (eql (qvalue= (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'P84)
         (eql (qdir (qval 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'DEC)
         (solenoid-A-energized)
         (eql (qmag-order (qmag (qval 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T state))
                          (convert-user-qmag 'T742
                                             (qspace 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E
-HV_T                                                                                    ))
                                             state))
                          (qspace 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T state))
              '-))
    (values T NIL))
   ((and (eql (qvalue= (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'P142)
         (eql (qdir (qval 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV state)) 'DEC)
         (solenoid-B-energized)
         (eql (qmag-order (qmag (qval 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T state))
                          (convert-user-qmag 'T742
                                             (qspace 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E
-HV_T
                                             state))
                          (qspace 'PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T state))
              '-))
    (values T NIL))
   ))


(defun Open-HP-valve (state)
  ; LP may be open at transition time.  In that case LP is closed implicitly.
  ;(format *qsim-report* "~&~%Opening HP valve and closing LP valve...")
  (create-transition-state :from-state state
                           :to-qde (build-qde (new-configuration state 'o-hp-valve)
                                              :QDE-clauses '(
                                                             ,pneumatic-system-print-nam
es
                                                             ,(new-transitions state 'o-
hp-valve)
                                                             ,pneumatic-system-other
                                                             ))
                           :assert '(
                                (PNEUMATIC-SYSTEM_COMPOSED_ENGINE.QH@E-HV (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_ENGINE.QL@E-LV (0 nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.QIN@E-LV (0 nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.QOUT@HV-LV-R (0 nil)
)
                                (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF ((minf 0) nil)
)
                                (PNEUMATIC-SYSTEM_COMPOSED_HP-VALVE.QOUT@HV-LV-R (nil ni
l)); just opens
                                ; Qin is set to "inc". (continuous transition)
                                (PNEUMATIC-SYSTEM_COMPOSED_HP-VALVE.QIN@E-HV (nil inc))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.QIN@HV-LV-R (nil ni
l))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.QOUT@R-P (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.P (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.PQ (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.DPOUT (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.Fs ((0 inf) nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.R (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.QHI@R-P (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.QHO@P-L (nil nil))
                                ; Error is set to "dec" to constraint excessive branchin
g.
                                ; (Fast Controller)
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.ERROR (nil dec))
                                ; TCO is set to "dec" to constraint excessive branching.
                                ; (Fast Controller)
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.TCO (nil dec))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.DTC (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.HFX (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.dQCI ((minf inf) nil
)
                                (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.DT (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED_LOAD.Q@P-L (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@HV-LV-R (P12 nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@R-P (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@HV-LV-R_T
                                        (,(new-effort@hv-lv-r_t state 'o-hp-valve) nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@R-P_T (nil nil))
                                (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@P-L_T (nil nil))
                                )
                           :inherit-qmag :rest
                           :inherit-qdir NIL
                           :text "Open HI Pressure Valve"))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;  SYSTEM CONFIGURATIONS  and TRANSITION CLAUSES  ;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(define-configuration HP-O-LP-C
  Pneumatic-System "HP Valve OPEN and LP Valve CLOSED"
  (HP-Valve (impl primitive) (mode open))
  (LP-Valve (impl primitive) (mode closed))
  )

(defconstant HP-O-LP-C-transitions
  '(transitions
    (Close-HP-VALVE-T Close-HP-VALVE)
    (Open-LP-VALVE-T Open-LP-VALVE)
    ))

;;;

(define-configuration HP-C-LP-O
  Pneumatic-System "HP Valve CLOSED and LP Valve OPEN"
  (HP-Valve (impl primitive) (mode closed))
  (LP-Valve (impl primitive) (mode open))
  )

(defconstant HP-C-LP-O-transitions
  '(transitions
    (Open-HP-VALVE-T Open-HP-VALVE)
    (Close-LP-VALVE-T Close-LP-VALVE)
    ))

;;;

; Cannot happen.

(define-configuration HP-O-LP-O
  Pneumatic-System "HP Valve OPEN and LP Valve OPEN"
  (HP-Valve (impl primitive) (mode open))
  (LP-Valve (impl primitive) (mode open))
  )

(defconstant HP-O-LP-O-transitions
  '(transitions
    (Close-HP-VALVE-T Close-HP-VALVE)
    (Close-LP-VALVE-T Close-LP-VALVE)
    ))

;;;

(define-configuration HP-C-LP-C
  Pneumatic-System "HP Valve CLOSED and LP Valve CLOSED"
  (HP-Valve (impl primitive) (mode closed))
  (LP-Valve (impl primitive) (mode closed))
  )

(defconstant HP-C-LP-C-transitions
  '(transitions
    (Open-HP-VALVE-T Open-HP-VALVE)
    (Open-LP-VALVE-T Open-LP-VALVE)
    ))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  QDE TRANSITION TABLE  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; THE TABLE ;;;

; 2 nested alists ((cur-QDE ((trans new-QDE) ...)) ...)

(defconstant TRANSITION-TABLE
  '(
     (HP-C-LP-C_PNEUMATIC-SYSTEM_COMPOSED ((O-HP-VALVE HP-O-LP-C)
                                           (C-HP-VALVE NIL)
                                           (O-LP-VALVE HP-C-LP-O)
                                           (C-LP-VALVE NIL)))
     (HP-O-LP-C_PNEUMATIC-SYSTEM_COMPOSED ((O-HP-VALVE NIL)
                                           (C-HP-VALVE HP-C-LP-C)
                                           (O-LP-VALVE HP-O-LP-O)
                                           (C-LP-VALVE NIL)))
     (HP-C-LP-O_PNEUMATIC-SYSTEM_COMPOSED  ((O-HP-VALVE HP-O-LP-C)       ; NOT HP-O-LP-O
!
                                            (C-HP-VALVE NIL)
                                            (O-LP-VALVE NIL)
                                            (C-LP-VALVE HP-C-LP-C)))
     (HP-O-LP-O_PNEUMATIC-SYSTEM_COMPOSED  ((O-HP-VALVE NIL)
                                            (C-HP-VALVE HP-C-LP-O)
                                            (O-LP-VALVE NIL)
                                            (C-LP-VALVE HP-O-LP-C)))
     ))

;;; THE HANDLERS ;;;

(defun new-configuration (state transition)
  (cond
   ((cadr (assoc transition (cadr (assoc (qde-name (state-qde state)) TRANSITION-TABLE))
)))
    (T
     (cerror "Perform transition, in current QDE" " Unexpected transition ~A while in ~A"
             transition (qde-name (state-qde state)))
     (qde-name (state-qde state)))))


(defun new-transitions (state transition)
  (let ((conf-name
          (cadr (assoc transition (cadr (assoc (qde-name (state-qde state)) TRANSITION-TA
BLE)))))
         tr-clause)
        (setq tr-clause
              (symbol-value (find-symbol (concatenate 'string (string conf-name) "-TRANSI
TIONS"))))
        (if (null tr-clause)
            (cerror "Assume NIL clause"
                    " No transitions-clause for QDE ~a or unexpected transition ~a while
in ~a"
                    conf-name transition (qde-name (state-qde state))))
       tr-clause))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  SYSTEM CONTROL STRUCTURE  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defmacro system-status (control)
  `(cadr (assoc ',control PNEUMATIC-CONTROLS)))


(defun solenoid-A-energized ()
  (COND
   ; PSCS AUTO & WAIR OFF
   ((AND (eql 'AUTO (system-status PSCS))
         (eql 'OFF (system-status WAIR))))
   (T
    NIL)))

(defun solenoid-B-energized ()
  (COND
   ; PSCS AUTO & WAIR ON
   ((AND (eql 'AUTO (system-status PSCS))
         (eql 'ON (system-status WAIR))))
   ((eql 'HI (system-status PSCS)))
   (T
    NIL)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  PRINT NAMES  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq pneumatic-system-print-names
  '(print-names
    (PNEUMATIC-SYSTEM_COMPOSED_ENGINE.TR "Throttle, Eng." TR)
    (PNEUMATIC-SYSTEM_COMPOSED_HP-VALVE.QIN@E-HV "Flow In, HP V." Q)
    (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.QIN@E-LV "Flow In, LP V." Q)
    (PNEUMATIC-SYSTEM_COMPOSED_LP-VALVE.PDIFF "P. Diff., LP V." P)
    (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.QIN@HV-LV-R "Flow In, PR." Q)
    (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.X "Valve Op., PR." X)
    (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.P "P. drop, PR." P)
    (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.R "Resist., PR." R)
    (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.PQ "P. from Q, PR." P)
    (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.FS "Spring F., PR." F)
    (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.DX "V. speed, PR." DX)
    (PNEUMATIC-SYSTEM_COMPOSED_REGULATOR.DPOUT "D/Dt Pout, PR" P)
    (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.QHI@R-P "Ho Flow In,  PC." Q)
    (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.QCI "Co Flow In, PC." Q)
    (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.TCO "Co T. Out, PC." T)
    (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.HFX "Heat Xchg., PC." H)
    (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.DT "DT, PC." T)
    (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.ERROR "Error, PC." T)
    (PNEUMATIC-SYSTEM_COMPOSED_PRECOOLER.dQCI "d/dt QCI, PC." Q)
    (PNEUMATIC-SYSTEM_COMPOSED_LOAD.Q@P-L "Flow In, Load" Q)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@HV-LV-R "P. In, PR." HIP)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@R-P "P. Out, PR." RP)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV "HI Stg. P." HIP)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-LV "LO Stg. P." LOP)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@P-L "P. at Load." PL)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@P-L_T "T. at Load." T)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@R-P_T "Ho T. In, PC." T)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-P_T "Co T. In, PC." T)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T "HI stg. T." T)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-LV_T "LO stg. T." T)
    (PNEUMATIC-SYSTEM_COMPOSED.EFFORT@HV-LV-R_T "T. In, PR." T)
    ))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  OTHER GENERAL CLAUSES  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defconstant pneumatic-system-other
  '(other
    (qspace-hierarchy
     ((PNEUMATIC-SYSTEM_COMPOSED.EFFORT@HV-LV-R
       PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-LV
       PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV
       ) ->
      (*seq 0 (*eqv PLO* PIN*)  PHI* P12 (*eqv PLO+ PIN+) P84 P142 inf))
     ((PNEUMATIC-SYSTEM_COMPOSED.EFFORT@HV-LV-R_T
       PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-LV_T
       PNEUMATIC-SYSTEM_COMPOSED.EFFORT@E-HV_T
       PNEUMATIC-SYSTEM_COMPOSED.EFFORT@R-P_T) ->
      (*seq 0 Tamb* TLO* THI* inf))
     ((PNEUMATIC-SYSTEM_COMPOSED.EFFORT@R-P
       PNEUMATIC-SYSTEM_COMPOSED.EFFORT@P-L
       ) ->
      (*seq 0 P46 PP inf))
    )))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  DEBUG UTILITIES  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun sqs (state)
  (dolist (qs (state-qspaces state))
          (format *qsim-report* "~a~%" qs)))

(defun sqv (state)
  (dolist (qs (state-qvalues state))
          (format *qsim-report* "~a~%" qs)))

(defun res ()
  (reset-trace-switches))

(defun tr ()
  (setq trace-constraint-filter t trace-tuples t))

(defun pron ()
  (setq *image-disposal* :both))

(defun proff ()
  (setq *image-disposal* :screen))
```