1995113219 N95-19635 1

57010

P- 11

QPA-CLIPS: a language and representation for process control

Thomas G. Freund*

Abstract

QPA-CLIPS is an extension of CLIPS oriented towards process control applications. Its constructs define a dependency network of process actions driven by sensor information. The language consists of 3 basic constructs: TASK, SENSOR and FILTER. TASKs define the dependency network describing alternative state transitions for a process. SENSORs and FILTERs define sensor information sources used to activate state transitions within the network. *deftemplate*'s define these constructs and their run-time environment is an interpreter knowledge base, performing pattern matching on sensor information and so activating TASKs in the dependency network. The pattern matching technique is based on the repeatable occurrence of a sensor data pattern. QPA-CLIPS has been successfully tested on a SPARCStation providing supervisory control to an Allen-Bradley PLC 5 controller driving molding equipment.

1.0 Introduction - the need

Process control is the science and, at times, art of applying and holding the right setpoint value and/or mixing the right amount of an ingredient at the right time. But, when is the time right ? And, if we know enough about the material or mixture, what is the right setpoint value and/or amount that has to be mixed ?

Materials scientists and engineers have spent years of painstaking experimentation and analysis to characterize the behavior of metals, plastics, and composites. Along the same vein, manufacturing engineers and factory floor operators have developed many person-years worth of practical "know-how" about material behavior under a variety of processing conditions.

In addition, though there is always room for improvement, significant strides have been made in useful sensor technology for acquiring information on material behavior. As the types of materials used in everyday products increase in complexity, the demand increases for embedding a better understanding about material behavior directly into the control of their manufacturing processes.

* Author's address - Pratt & Whitney, 400 Main Street, Mail Stop 118-38, East Hartford CT 06108. This work was supported by a grant from the National Center for Manufacturing Sciences, Ann Arbor, MI 48108 and in cooperation with Allen-Bradley Co. (Milwaukee, WI) and Erie Press Systems (Erie, PA)

QPA-CLIPS, the language and its run-time environment, is a means to directly tie our best understanding of material behavior to the control of material forming or curing processes. This is accomplished by intelligent mapping of sensor information on material behavior to changes in process parameter values. The changed parameter values, in turn, are used to directly drive process equipment .

2.0 The problem

Material forming processes transform a prescribed volume of material(s) into a desired net shape. In curing, material is processed in a way that changes material properties to a set of desired characteristics. In either case, the proper choice of tooling along with a properly defined process cycle, or "recipe", are crucial to the consistent production of quality parts. Variations in material behavior from an expected norm must be accounted for in both the initial conditions of the material and, particularly, the cycle or "recipe" driving the process equipment.

2.1 Controlling a forming or curing process

Attaining a desired final form and material characteristics in forming or curing processes requires effective control of force and heat application. Control of force involves application of mechanical pressure to distort or redistribute a set volume of material. within a constraining volume, defined by the tooling (i.e., dies) used in the process.

In the case of forming processes, the material usually requires to be more pliable than its natural state at room temperature. Heat application then becomes an integral part of the pressure application. For curing processes, the application of heat itself, with some pressure being applied in certain cases, is at the core of the processes. In either case, heat control, then, must be *synchronized* with force control and the state of the material must be monitored in-process to determine the right point in the process cycle where heat and/or force application is needed.

One must also take into account the physical limitations imposed by the machinery used for the process. Repeated extreme rates of heat or force application can lead to frequent machine breakdowns and, consequently, make the process economically undesirable.

2.2 "Listening" to the material

In-process monitoring is not the act of collecting streams of historical data for off-line analysis; though this is an important step in creating effective process control. When we collect information during a dialogue, we don't necessarily capture every single word and nuance. Rather, we normally record essential points and supporting information that capture the theme or goal of what is being exchanged.

In a similar way, analyzing process data involves the ability, as in listening, to differentiate between steady or normal process behavior and patterns , or *events,* indicating the onset of a change in material state.

As a simple example, **Figure 1** shows a plot of sensor data over time during a curing process. Point A indicates the onset of the peak, point C, in the sensor data; while B indicates that we are past the region around C. If C is indicator of an optimum material condition for heat or force application, understanding data patterns in A and/or B can be used to trigger that application.

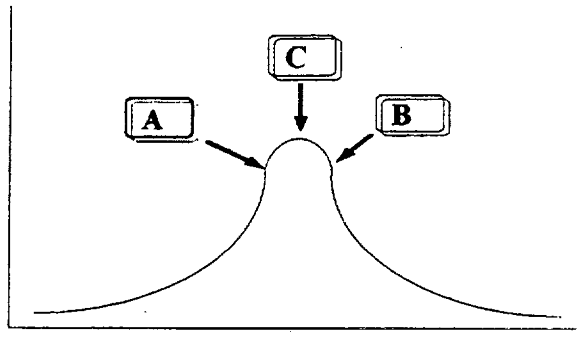

Figure 1

So that, creating a process "recipe" becomes a matter of identifying those key patterns or events, understanding their relationship to heat or force application, and finally linking that application to the occurrence of these events.

3.0 A solution : QPA-CLIPS

QPA-CLIPS, or Qualitative Process Automation CLIPS, is a language with a supporting run-time environment used to describe and run process "recipes" as a set of causal linkages between a specific sensor event(s) and application of a heat or force. The concept of qualitative process analysis was developed as a technique for intelligent control based on interpretation of sensor information [1]. QPA-CLIPS is an implementation of this concept in the CLIPS environment.

3.1 Architecture

The execution model of QPA-CLIPS is basically a traversal across a dependency network. The nodes in that network describe one or more sensor events, their related application actions, and one or more pre-conditions which must be satisfied in order to activate or traverse the node.

Traversal through a node consists of 3 phases:

(1) satisfaction of pre-conditions,

(2) detection of sensor events,

(3) performing required heat or force application.

Transition to the next phase cannot occur until successful completion of the current phase. Once phase 3 is completed, one of the remaining nodes in the network is chosen for traversal using the same 3-phase method. If all nodes have been traversed, traversal is halted. Pre-conditions are either traversal through another node, unconditional (i.e. START node), or the successful diagnostic checks on a sensor. The flow chart in **Figure 2** summarizes the traversal process.

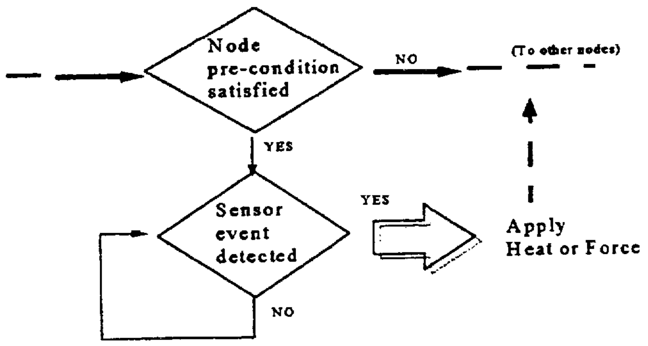

Figure 2

Node traversal is implemented as an interpreter knowledge base exploiting the underlying CLIPS search methods.

There are currently 3 constructs making up the QPA-CLIPS language: TASK, SENSOR, and FILTER. They are defined within the interpreter knowledge base through a set of *deftemplate* 's [2] . The contents of a node are described by the TASK. SENSOR defines sources of raw sensor data; while FILTER applies mathematical functions to SENSOR data. A collection of TASKs describing a dependency network, along with the required SENSORs and FILTERs, is referred to as a *process model* and is consistent with the GRAFCET standard based on Petri Nets [3].

3.1.1 TASK

TASKs encapsulate the pre-conditions for node traversal, sensor event descriptions, and the prescribed heat or force application. In BNF notation form, this translates to:

```
( TASK
    (name <name>)
    (start-when  <pre-condition>)
    (look-for <sensor-event>)
    (then-do <application>) )
```

<name> is a unique label used whenever another TASK refers to this TASK throughout the process model. <pre-condition> is a string type that can be either:

```
    <task-name> COMPLETE,
or
    <sensor> OK
or
    START
```

In the first form, the pre-condition becomes the completion or traversal through another node, or TASK. The second form checks completion of a diagnostic check for a sensor. Sensor diagnostics are not currently an explicit construct in QPA-CLIPS. Rather, they are embedded in the sensor I/O functions integrated with CLIPS. The third form of <pre-condition> is makes that TASK the initial traversal point in the dependency network.

<sensor-event> is a string type that can have multiple occurrences of the form:

```
        <source> <direction> <nominal> <tolerance> <repeat>
```

<source> is the label for a SENSOR or FILTER providing data identifying this event. <direction> can be RISES, DROPS, or HOLDS. The remaining items are numbers describing respectively a threshold value that must be reached, the tolerance band around the nominal value, and the number of times that SENSOR or FILTER data must meet this threshold-and-tolerance- band occurrence without interruption.

So for example, the sensor event clause of

```
    flow RISES 50 0.01 3
```

translates to data from SENSOR flow must rise at least 0.01 above 50 cc/min. for 3 consecutive times in order to be identified as this sensor event.

<application> is of the form :

```
        <parameter> <amount>
```

where <parameter> is a label for a process parameter and <amount> is a new value or setting for that parameter. An example is :

```
        PR  5
```

or set pressure to 5 tons. Application of heat or force then is basically a change in the value of a memory location which, in turn, drives the process.

3.1.2 SENSOR

SENSORs are one of 2 possible sources of data for a sensor event defined within a TASK and defined as follows:

```
( SENSOR
   (name <name>)
   (max-allowed <value>)
   (min-allowed <value>) )
```

<name> is a unique label used whenever a TASK or FILTER refer to this SENSOR throughout the process model. *max-allowed* and *min-allowed* provide the allowable range of values for the sensor data. As an example, a flowmeter within a process model can be defined as:

```
( SENSOR
   (name flowmeter)
   (max-allowed 100)
   (min allowed 5) )
```

The rate of the flowmeter can range between 5 and 100 cc/min. Currently , retrieval of sensor data is accomplished through user-defined functions embedded within CLIPS. Association between a SENSOR and these functions is performed through the QPA-CLIPS interpreter. The implementation of SENSOR assumes use of one sensor for the process. This suffices for current applications of QPA-CLIPS. However, for multiple sensor input, the SENSOR construct will be modified to include a reference to a specific function, referred to a the SENSOR source, or:

```
( SENSOR
   (name <name>)
   (max-allowed <value>)
   (min-allowed <value>)
   (source <function>) )
```

3.1.3 FILTER

FILTERs are the other source of data for a sensor event and is defined as:

```
( FILTER
   (name <name>)
   (max-allowed <value>)
```

```
    (min-allowed <value>)
    (change <sensor>)
    (using <formula>) )
```

<name> is a unique label used by a TASK whenever referring to this FILTER in order to identify a sensor event. Both *max-allowed* and *min-allowed* are used in the same manner as in SENSOR. But, here, they refer to the calculated value created by this FILTER. <sensor> is the label referring to the SENSOR supplying raw sensor data to this FILTER. <formula> is a string describing the combination of mathematical functions used to translate the raw sensor data. so, for example, a formula string of "SLOPE LOG" is the equivalent of :

d(log S)/dt,   where S is the raw sensor data.

Future enhancements to the FILTER construct will be the ability for expressing formulas in more natural algebraic terms. So, "SLOPE LOG", for example, will take on a form like dLOG / dt.

3.2 Building a process model

The first step in developing a process model is a through description of how the process runs, what information can be extracted about the process during a cycle run, what sensors are available to extract this information, and how is the sensor information related to application of heat and/or force to the material. These relationships can be investigated through techniques such as Design of Experiments or Taguchi methods. Once they are verified, a dependency network can be build from these relationships in order to specify alternative paths for running the process cycle; depending on the sensor data patterns being extracted. This network can then be encoded as a process model.

As a simple example, a molding process for a panel in a cabinet enclosure is now switching to a new material, requiring that pressure be triggered based on material temperature. For this material, a constant heat setting must be applied throughout the cycle. A sensor, called a thermowidget, was selected to retrieve in-process material temperature. Its operating range is 500°F and 70°F. So, its SENSOR definition is:

```
( SENSOR
   (name thermowidget)
   (max-allowed 500)
   (min-allowed 70))
```

Two pressure applications are required by the process: an initial setting to distribute material throughout the mold and a final setting to complete the part. The initial setting must be applied when the material temperature reaches a value of 100°F and the final setting when the rate of change of the material temperature is 0. We first need a FILTER to define the slope of the temperature or:

```
( FILTER
   (name thermoslope)
   (max-allowed 10)
   (min-allowed -10)
   (change thermowidget)
   (using "SLOPE") )
```

We achieve the pressure application through two TASKs:

```
( TASK
    (name first-setting)
    (start-when  "START")
    (look-for "thermowidget RISES 100 0.1 5")
    (then-do "PR 2") )

( TASK
    (name final-setting)
    (start-when  "first-setting COMPLETE")
    (look-for "thermoslope DROPS 0 0.01 3")
    (then-do "PR 5") )
```

Another TASK can be added to complete the cycle and release the finished part.

4.0 Run-time environment

A process model is executed through a QPA-CLIPS interpreter: a CLIPS knowledge base consisting of the *deftemplate*'s defining the TASK, SENSOR, and FILTER constructs, a rule set which carries out the interpretation cycle, and *deffunction*'s supporting the rule set. Underlying this knowledge base are the user-defined functions integrating the process model with sensors and process equipment.

4.1 The Interpreter

At the core of the interpreter are a set of rules which cycle through the currently available TASKs in a process model and implement the 3-phase method for node traversal mentioned above. When the QPA-CLIPS environment along with the appropriate process model is invoked, control transfers to the TASK containing the pre-condition of START. From that point on, the node traversal method takes over. The QPA-CLIPS interpreter knowledge base can be ported to any CLIPS-compatible platform.

4.2 Sensor and controller integration

Sensors are integrated through user-defined functions embedded within CLIPS. They currently interact with sensor hardware through serial (RS-232-C) communications. These functions are the only platform dependent components within QPA-CLIPS.

Integration of heat and force application is done through a memory mapping paradigm between sensor events and parameter settings where TASK actions, as defined in the

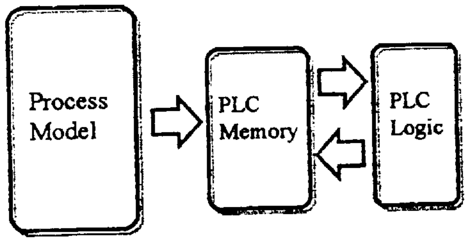

Figure 3

then-do clause, are mapped to the memory location of another control system (i.e. a PLC) which, in turn, directly drives process machinery (see Figure 3). These new settings then trigger signals to heaters or motors driving the equipment.

5.0 Current status

A version of QPA-CLIPS has been demonstrated on a composite molding application. The system consists of a SPARCStation 2 running QPA-CLIPS linked to an Allen-Bradley PLC5/40 driving a molding press. It has successfully created a number of production quality parts for a jet engine.

6.0 Future enhancements

Though the current version of QPA-CLIPS has been demonstrated to work successfully, several opportunities for enhancements have been mentioned in the description of its constructs. In addition, other opportunities for enhancements exist in the development and run-time environments of QPA-CLIPS.

6.1 A GUI for QPA-CLIPS

Currently, a process model is created through the use of a text editor, such as emacs or textedit. The completed model is tested on a simulated run-time environment. Taking a cue from the visual development tools such as those found under Microsoft Windows, a graphical development environment for a process model will greatly ease the development of a process model. The syntax of the QPA-CLIPS constructs are simple enough for process engineers to work with. Nevertheless, as process models grow in complexity, a need will arise for easy-to-use model navigation and debugging tools seamlessly connected to the simulated run-time environment.

6.2 Automating process model creation

Experience with developing process models has shown that the bulk of the time is actually spent in experimentation; trying to establish the relationships between sensor data and heat/force application points. What is needed to streamline this aspect of the model creation process is the partial or complete automation of the experimentation phase leading to the automatic creation of a process model; since a new material system will require a new process model. The work on DAT-GC by Thompson et al [4] offer a good start in that direction.

6.3 Exploiting fuzzy linguistics

In the definition of the *look-for* clause of a TASK construct, one needs a rather accurate description of not only the goal or threshold value to be reached, but also a tolerance band around that value as well as a repeatability count on the number of times in a row that the event must appear. There are cases where a less exacting or *fuzzy* [5] description will suffice. So, using our example model from 3.2 above, the *look-for* clause for the final pressure application can then be:

```
(look-for "thermoslope dropping IMMEDIATELY to ZERO")
```

IMMEDIATELY is a fuzzy term similar to the repeatability count. ZERO, on the other hand, describes a fuzzy term associated with a range of precise values for a SENSOR or FILTER, which can be described with a new QPA-CLIPS construct such as:

```
( BEHAVIOR
   (name <name>)
   (for <source>)
   (range <value-list>) )
```

where <name> is the fuzzy term, <source> is the label for the SENSOR or FILTER, and <value-list> contains the values defining the membership function [5] for this BEHAVIOR.

6.4 Exploiting parallelism

It goes without saying that node traversal within a process model is inherently a method that can easily converted to parallel processing. Having a parallel version of CLIPS operating in an affordable parallel processing platform will enable QPA-CLIPS to easily operate in a process control scenario requiring multiple sensors.

7.0 Conclusions

QPA-CLIPS, an extension of CLIPS for process control, is a proven tool for use by process engineers in developing control models for forming or curing processes. Its simple syntax and integration into CLIPS provides a powerful, yet straightforward, way to describe and apply control of a process driven by sensor events, or distinct patterns in sensor data. Several enhancements have been suggested and currently strong interest exist in the automatic generation of process models and fuzzifying the description of sensor events.

References

[1] Park, Jack, **Toward the development of a real-time expert system**, 1986 Rochester FORTH Conference Proceedings, Rochester NY, June 1986, pp. 23-33

[2] CLIPS Reference Manual, Vol. I, **Basic Programming Guide**, CLIPS Version 5.1, Sept. 1991

[3] David, Rene and Hassane Alla, **Petri nets for modeling of dynamic systems - a survey**, *Automatica*, Vol. 30, No. 2, pp.175-202, 1994

[4] Levinson, Richard, Peter Robinson, and David E. Thompson, **Integrated perception, planning and control for autonomous soil analysis**, Proc. CAIA-93

[5] Terano, Toshiro, Kiyoji Asai, and Michio Sugeno, **Fuzzy Systems Theory and its applications**, Academic Press, 1992