For: karonis
Printed on: Tue, Nov 2, 1993 15:51:41

Document: String Resolution
Last saved on: Tue, Sep 28, 1993 10:30:53

A Methodology For String Resolution

Nicholas T. Karonis
November 1992

1 Introduction

In this paper we present a methodology, not a tool. We present this methodology with the intent that it be adopted, on a case by case basis, by each of the existing tools in EPICS.

In presenting this methodology, we describe each of its two components in detail and conclude with an example depicting how the methodology can be used across a pair of tools.

2 Motivation

The task of any control system is to provide access to the various components of the machine being controlled, for example, the Advanced Photon Source (APS). By access, we mean the ability to monitor the machine's status (reading) as well as the ability to explicitly change its status (writing).

2.1 Importance of Names

The Experimental Physics and Industrial Control System (EPICS) is a set of tools, designed to act in concert, that allows one to construct a control system. EPICS provides the ability to construct a control system that allows reading and writing access to the machine. It does this through the notion of databases.

Each of the components of the APS that is accessed by the control system is represented in EPICS by a set of named database records. Once this abstraction is made, from physical device to named database records, the process of monitoring and changing the state of that device becomes the simple process of reading and writing information from and to its associated named records.

Databases are the fundamental building blocks in EPICS. All the tools in EPICS are designed to ultimately interact with the named records in the database. Named database records is a critical assumption that spans across all the tools in EPICS. It is the names of these records, and nothing else, that permits the various tools in EPICS to function in harmony. In other words, without the name of the database records, records that were created with one tool would not be able to be referenced by another tool.

2.2 Redundancy in the APS

One of the characteristics of the APS is its redundancy with respect to its physical devices. For example, the storage ring of the APS is composed of several sectors. The devices found in each of the sectors are virtually identical. As a result, the structure of the databases that control each of the sectors are also virtually identical. The major difference is only in the names of the database records.

2.3 Characteristics of a Good Methodology

We have seen two characteristics in EPICS and the APS. The first is the assumption that all database records are named, and it is that assumption that binds all the tools in EPICS. The second is the redundant nature of the physical devices that compose the APS.

A good methodology must harness the unifying power found in the assumption of named records as well as exploit the redundancy found in the APS. We believe this methodology does both.

3 String Resolution

As a methodology, string resolution is not limited to database record names. It is a methodology that can be applied to any set of strings. This broad application base proves to be essential when applying the methodology across all existing tools, and we believe will be most helpful in adopting the methodology to future tools.

String resolution is composed of two phases, substitution and enumeration.

3.1 Substitution

Substitution is the process of transforming one string of characters into another.

Definition: Given a finite alphabet Σ, s is a *string of* Σ, or a *string*, if and only if $s \in \Sigma^*$.

The transformation is guided by a user–supplied transformation function.

Definition: Given a finite alphabet Σ, function f is a *transformation function* if and only if:
1) (string conversion) f: $\Sigma^* \rightarrow \Sigma^*$ and
2) (grounding) $f(s) = f(f(s))$.

Definition: Given a transformation function f, a string s is *grounded with respect to* f, or *grounded*, if and only if $s = f(s)$.

Transformation functions transform strings to grounded strings.

3.1.1 Specifying Transformation Functions

Transformation functions can be specified with user–supplied ASCII files. Here is an example file.

<A> –> XY

This ASCII file represents a valid transformation function. It specifies that every occurrence of <A> in the source string be converted to XY. For example, if s = abc<A>def<A>, then $f(s)$ = abcXYdefXY. Note that *all* occurrences of <A> are transformed in a single invocation of the transformation function, thus preserving grounding.

It is possible for for the ASCII file to be more complex. Here is another example file.

<A> –> X<B>Z
<B> –> Y

If s = a<A>b<B>, then f(s) = aXYZbY, again preserving grounding.

We have introduced the ability for elements on the right of an arrow to appear as the token for the left side of some other arrow in the same ASCII file. This allows for the possibility of cyclical specifications. Here is an example of an ASCII file with a cyclical specification.

<A> –> X<B>Z
<B> –> Y<C>
<C> –> <A>

Note the cycle encountered when attempting to transform s = x<A>y. The detection of such cycles in an ASCII file is a simple matter. Those ASCII files that contain cycles are invalid and cannot be used to specify a transformation function. The acyclic property of an ASCII file is essential to insure the grounding property of the resulting transformation function.

3.1.2 Using Transformation Functions

Once the transformation function has been specified by a valid ASCII file it is ready for use. Selected strings from the input from an EPICS tool should be processed by the transformation function. Because the transformation functions always result in grounded strings, this process need only be done once.

3.2 Enumeration

The process of enumeration works with two elements, a finite tree and an enumeration list. It is assumed that all tools in which enumeration is to be used can have its input represented as a finite tree. Given a finite tree and an enumeration list, the enumeration process generates a new finite tree that serves as input to the EPICS tool.

3.2.1 The Enumeration List

Here is the general structure of each element in an enumeration list,

id = enumeration set

where id is a unique enumeration id token and enumeration set is a non–empty finite set. As a set, there is no order on the elements in the enumeration set and there are no duplicates.

3.2.2 The Enumeration Process

Given a finite tree and an enumeration list, the enumeration process entails performing a pre–order traversal on the tree processing each node accordingly. Those nodes that do not have any enumeration id tokens are left unchanged. The other nodes, those with at least one enumeration id token, must be expanded.

Each enumeration id token has an associated enumeration set. Expanding the node requires taking the Cartesian product of all the associated enumeration sets and creating a new node for each element in the Cartesian product. Each newly created node has effectively instantiated each of the enumeration id tokens with an element from its enumeration set.

The entire subtree of the node being processed is replicated as subtrees of each of the newly created nodes. For each new subtree rooted at the newly generated nodes, all enumeration id tokens that appeared in the node being processed are replaced with the instantiated value of their root.

The node being processed, and all its children, are removed from the tree and replaced by all its newly created subtrees.

4 Example

We examine the methodology as it applies to two tools in EPICS, the database and the alarm handler. String substitution is straightforward and is not covered in this example. This example concentrates on the enumeration process over these two tools.

In this example we consider a magnet. We assume that it requires two database records to control a magnet, r1 and r2, where r1 forward links to r2. We assume further that there are two magnets per sector and two sectors per database.

4.1 Database

Here is the database.

```
r1m[m]s[s]
   FWD = r2m[m]s[s].VAL

r2m[m]s[s]
```

For purposes of enumeration, a database is considered a tree. A root node is artificially constructed and called database. Each record of the database is a child of the root node. Fields within a record represent the record's children.

Here is the enumeration list that conforms to the assumptions in our example; two magnets per sector and two sectors.

```
[s] –> {1, 2}
[m] –> {1, 2}
```

In processing the database tree with a pre–order traversal, the first node we encounter with enumeration id tokens is the record r1m[m]s[s]. After expanding that record and all its children we have the following database.

r1m1s1
   FWD = r2m1s1.VAL

r1m1s2
   FWD = r2m1s2.VAL

r1m2s1
   FWD = r2m2s1.VAL

r1m2s2
   FWD = r2m2s2.VAL

r2m[m]s[s]

Note that both enumeration id tokens [m] and [s] were fully expanded and that each instantiation of [m] and [s] was appropriately propagated to each of the fields (children) of the record (parent).

The enumeration process continues to the next node in the tree with an enumeration id token, the record r2m[m]s[s]. Here is the database after expanding that node.

r1m1s1
   FWD = r2m1s1.VAL

r1m1s2
   FWD = r2m1s2.VAL

r1m2s1
   FWD = r2m2s1.VAL

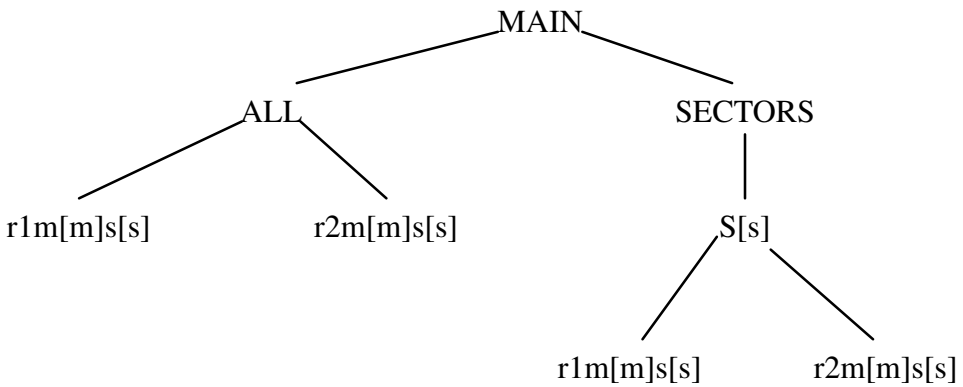
r1m2s2
   FWD = r2m2s2.VAL

r2m1s1
r2m1s2
r2m2s1
r2m2s2

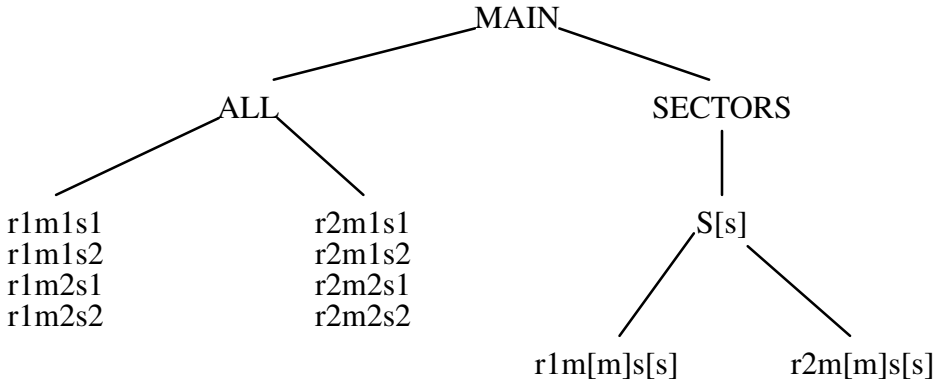
The resulting database has eight records in it. This was generated from the source database, having only two records, and the enumeration list.

4.2 Alarm Handler

Now consider generating an alarm configuration file for the alarm handler to monitor the channels in our example database. In our example, we chose to group all the channels in a single group called ALL as well as separating the channels by sector. In the following tree, all leaves are channels and all other nodes are groups.

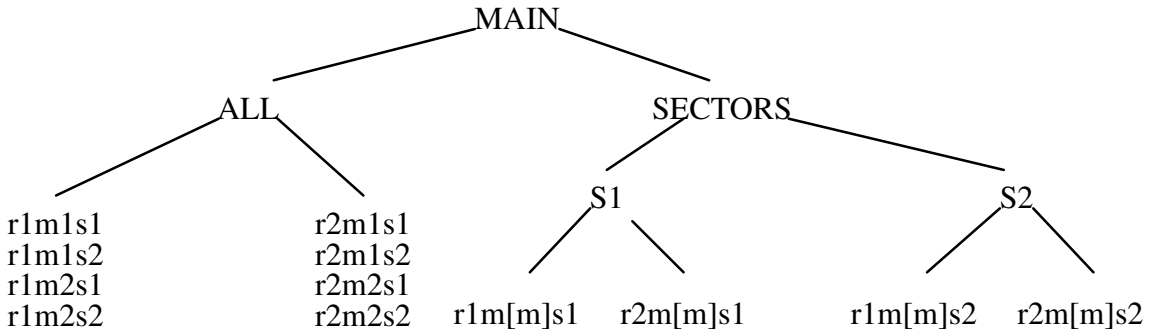


Using the *same enumeration list* we process this tree. Again, in a pre–order traversal, the first node we encounter with enumeration id tokens is the channel r1[m]s[s]. We expand it appropriately. Since it has no children, the expansion is straightforward. We proceed with the traversal and encounter the channel r2m[m]s[s] and expand it appropriately. Here is the tree after both expansions.



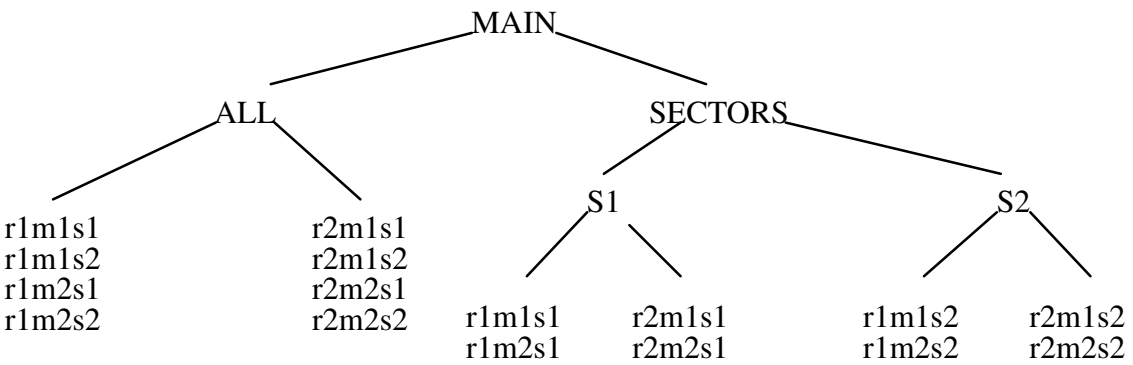
Although not clearly evident in the above diagram, the node ALL now has eight children. Each of the original single leaf nodes representing channels have been expanded to four leaf nodes, each representing single channels.

The pre–order traversal continues and we encounter node S[s].



Note that in instantiating the token [s] in the leaf nodes, the token [m] was left unchanged.

The final steps of the enumeration process expands the token [m] in each of the leaf nodes.



Given the same base alarm configuration file, we could have easily transformed the tree into one that monitors all the sectors by simply changing [s]'s enumeration set.