

## 1-1 Introduction to ASPL

ASPL, A Set Programming Languages, is a *setadic calculator*. The word *setadic* is introduced by the author and is specifically used herein to refer to a calculator whose symbolic operators are termed *setadic operators* and all of which perform algebraic set operations on datasets. These operators always precede their operands.

A **setadic calculator**, is a software appliance that takes structured data objects as input and breaks them into their constituent containment pathes so that they can be analyzed according to their structural components. ASPL maps algebraic data groups into containment pathes, and the appliance offers setadic operators to aggregate and to determine the relationships between their constituent elements. It renders them into a form suitable to be comprehended visually by the user before displaying their relationships on the user terminal.

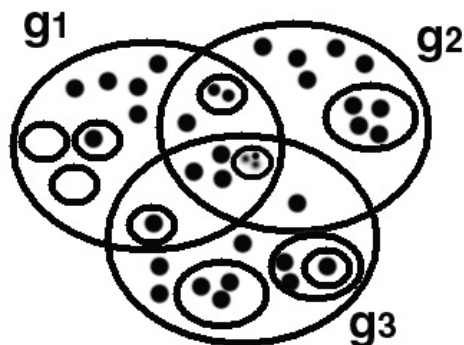
Therefore, ASPL is an interactive software appliance that is started on the UNIX shell command prompt. Once ASPL is installed on the computer system, the user can start using it interactively to do sophisticated set operations, or can run ASPL scripts on the system where ASPL is installed. The ASPL interpreter provides a powerful and intuitive interactive computing facility that shares some similarities to a classic calculator. However, ASPL operators are set operators, and its variables are set variables. For the operator using ASPL, all variables are typeless and the commands are short and simple mnemonics that are easy to remember. ASPL does not require a database or any third party libraries. The program runs on UNIX systems where the standard Perl interpreter is available.

Your ASPL interpreter can perform a large number of sophisticated set operations using the preprogrammed set operators. The calculation results may then be assigned to variables called *set variables*. The set variables are presented as arguments to set operators to do further calculations. In addition, when you start the ASPL interpreter you are assigned a named workspace. It is this named workspace that is used by the ASPL instance you started to save the results and the variables of all your calculations. ASPL uses a stack to save the results performed by its operations, and save the variables in internal symbol tables. ASPL variables are all global variables, and when persistence is enabled these global variables are all shared by users connecting to the same workspace; in which case if two or more ASPL instances are started with the same named workspace then all variables are being shared globally by all ASPL instances. The workspace directory should be available to all instances.

ASPL path containment can span multiple groups and subgroups along their elements. The following figure shows groups, subgroups, and their elements. There are three groups labeled g1, g2, and g3.

**Note: ASPL groups image to be further explained in next Chapter using RANONEBIT workspace**

[full view](#)



**Groups g1 g2 g3,  
their subgroups,  
and their elements.**

-F- Fig. 1.1.1 [THREE GROUPS, THEIR SUBGROUPS, AND THEIR ELEMENTS]  
[This figure shows three groups labeled g1, g2, and g3. Each group may contain

**one or more subgroups or subsets. The elements within the groups and their subgroups are marked with black dots.]**

ASPL © 2024 by Bassem W. Jamaledine

The labeled groups can be assigned to ASPL variables. For instance, v1, v2, and v3 are three ASPL variables that represent the groups g1, g2, and g3 respectively. ASPL setadic operators can then perform set operations on these variables. For instance to get the set union of all three groups: **gU v1 v2 v3**, to get their set intersection: **g& v1 v2 v3**, to get their set difference: **g\ v1 v2 v3**, and to get their set partitions: **gP v1 v2 v3**.

Notice the mnemonic of these four basic set operators: gU, g&, g\, and gP in which the first letter is a lower case 'g' depicting the subject of on which the operation will take place, and the second letter (either one of U, &, \, and P) depicting the operation type. These four operators do the following operations, respectively:

Instruction	Operation
gU v1 v2 v3	get the group union considering the subgroups and elements
g& v1 v2 v3	get the group intersection considering the subgroups and elements
g\ v1 v2 v3	get the group difference considering the subgroups and elements
gP v1 v2 v3	partition the groups, subgroups, and their elements

In a functional way these operations can also be further explained:

Instruction	Operation
gU v1 v2 v3	Union(v1, Union(v2,v3))
g& v1 v2 v3	Intersection(v1, Intersection(v1,v2))
g\ v1 v2 v3	Difference(v1, Difference(v2,v3))
gP v1 v2 v3	Partition(v1,v2,v3)

The subject denoted with the 'g' is to consider both the subgroups and elements in their respective groups. However, if the subject is only to do the operations on the subgroups then one can use dU, d&, d\ and dP. Therefore to get the set union of *the subgroups only* of all the three groups: **dU v1 v2 v3**, to get their set intersection: **d& v1 v2 v3**, to get their set difference: **d\ v1 v2 v3**, and to get their set partitions: **dP v1 v2 v3**.

One can look into the elements of the groups only (excluding the subgroups) by using fU, f&, f\ and fP. Therefore to get the set union of *the elements only* of all three groups: **fU v1 v2 v3**, to get their set intersection: **f& v1 v2 v3**, to get their set difference: **f\ v1 v2 v3**, and to get their set partitions: **fP v1 v2 v3**.

For ASPL this is done through its *set builder and the setadic operation*. For example, the intersection of the elements found in v1, v2, and v3, denoted with **f& v1 v2 v3** is to parse (or humanly read) the information left to right: the set operator comes first (hence the meaning of *setadic operator*) then followed by the identifiers (representing the labeled groups or datasets). This is something that the reader is accustomed to from basic schooling, and can be seen through ASPL setbuilder by typing **setbuilder gU v1 v2 v3** at the ASPL prompt:

```
aspl> setbuilder gU v1 v2 v3

QUOTIENT SET BUILDER

{gU v1 v2 v3} <=>  gU  v1  v2  v3

Detailed view:

{gU v1 v2 v3} <=>

gU  v1  v2  v3
|   |   |   +----> set-variable
|   |   +-----> set-variable
|   +-----> set-variable
+-----> get the subgroups and the elements union
```

Set builder syntax is read from left to right, or from bottom to top.  
All ASPL setops are setadic: they take a setop followed by set variables.

As for the intersection, one can type **setbuilder g& v1 v2 v3** at the ASPL prompt:

```
aspl> setbuilder g& v1 v2 v3

QUOTIENT SET BUILDER
```

```
{g& v1 v2 v3} <=> g& v1 v2 v3
```

Detailed view:

```
{g& v1 v2 v3} <=>
```

```

g&  v1  v2  v3
|   |   |   +----> set-variable
|   |   +-----> set-variable
|   +-----> set-variable
+-----> get the subgroups and the elements intersection

```

Set builder syntax is read from left to right, or from bottom to top. All ASPL setops are setadic: they take a setop followed by set variables.

Each element within a group or subgroup can be further described and characterized by a set of attributes that are attached to it, and comparing it to another element is therefore based on these attributes. In ASPL every element has a checksum attribute, and the language provides a feature to subordinate setadic operators with a conditional predicate. Repeating the previous operation `g& v1 v2 v3` but we want to get only these subgroups and elements that have different checksums:

### setbuilder `g&`c v1 v2 v3`

```
aspl> setbuilder g&`c~ v1 v2 v3
```

QUOTIENT SET BUILDER

```
{g&`c~ v1 v2 v3} <=> g&`c~ v1 v2 v3
```

Detailed view:

```
{g&`c~ v1 v2 v3} <=>
```

```

g& ` c~ v1  v2  v3
| | | | | +----> set-variable
| | | | +-----> set-variable
| | | +-----> set-variable
| | +-----> have different checksums
| +-----> such that
+-----> get the subgroups and the elements intersection

```

Here we used the acute backtick ``` followed by a predicate. ASPL refers to this backtick simply with the word 'tick' and many of ASPL set operators can be ticked with specific predicates. For example, in the above example the operator is ticked with the predicate saying "when checksums are different" `g&`c~`.

In general, we are also accustomed to the forward slash, called a stroke or the solidus symbol `/`, that may follow a set operator to depict a quotient relation, and in ASPL this is called *stroking a set operator*. In the following example, the group intersection operator (`g&`) is being stroked with the quotient relation `r3`. Notice that the stroke is directly followed by a tilde then the relation `r3`. `r3` has been defined by the user but expanded here as `frx=.*$,mtm~` (that is saying to match the elements with regular expression `.*$` and whose `mtime` attribute differ).

```
aspl> setbuilder g&/~r3 v1 v2 v3
```

QUOTIENT SET BUILDER

```
{g&/~r3 v1 v2 v3} <=> g&/frx=.*$,mtm~ v1 v2 v3
```

Detailed view:

```
{g&/~r3 v1 v2 v3} <=>
```

```

g& / frx=.*$,mtm~ v1  v2  v3
| | | | | +----> set-variable
| | | | +-----> set-variable
| | | | +-----> set-variable
| | | +-----> have different make times
| | +-----> file name regular expression
| +-----> stroking the Quotient Relation
+-----> get the subgroups and the elements intersection

```

The definition of `r3` was performed with the command shown below and at any time the user can type `q` to display the quotient relation table:

```
aspl> q r3 := frx=.*$,mtm~
aspl> q
```

Coded QR

qr	user	code
r3	(1)root	frx=.*\$,mtm~

This is equivalent to typing `g&`mtm~ v1 v2 v3` and in this particular case all the following three commands are equivalent:

```
aspl> g&/~r3 v1 v2 v3
aspl> g&/frx=.*$,mtm~ v1 v2 v3
aspl> g&`mtm~ v1 v2 v3
```

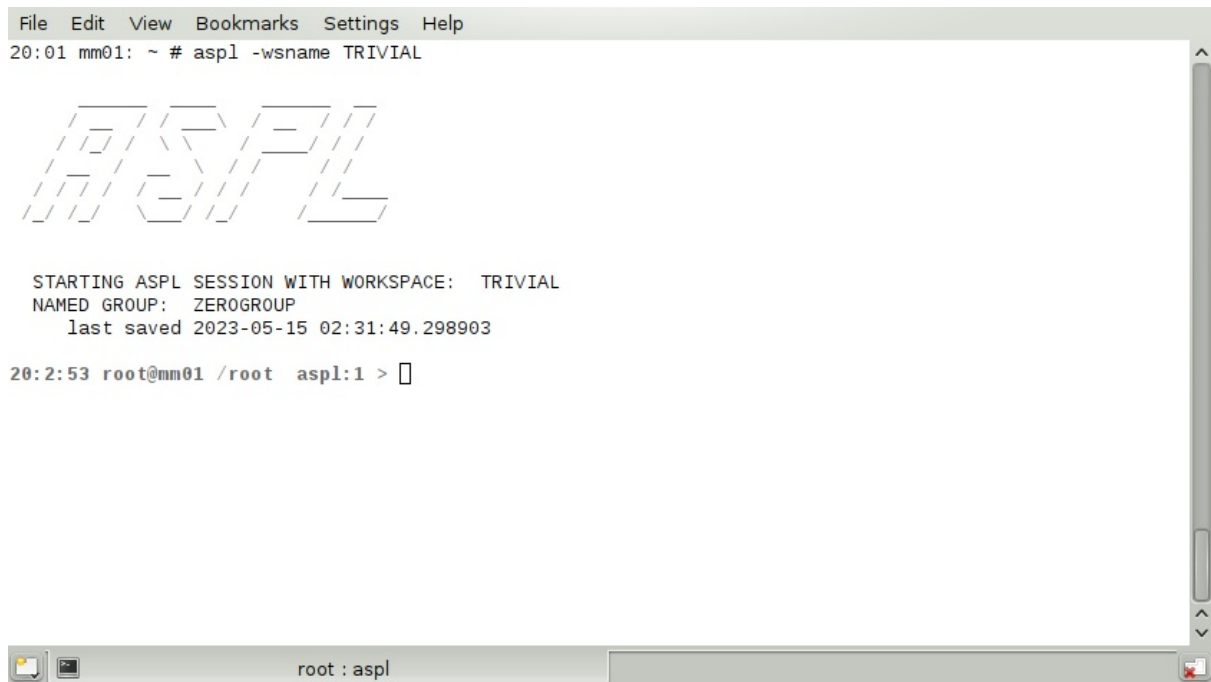
In this book the ASPL prompt is denoted with **aspl>** and the UNIX shell prompt is represented with the hash **#** symbol.

See appendix AAAAA on how to install ASPL on your UNIX system.

## ■ ASPL Building Blocks

**Note: ASPL Startup TRIVIAL ASPLv1.00-startup-trivial.png**

[full view](#)



**-F- Fig. 1.1.2 [ASPL Startup TRIVIAL][ASPL Starting with TRIVIAL workspace]**

*ASPL © 2024 by Bassem W. Jamaledine*

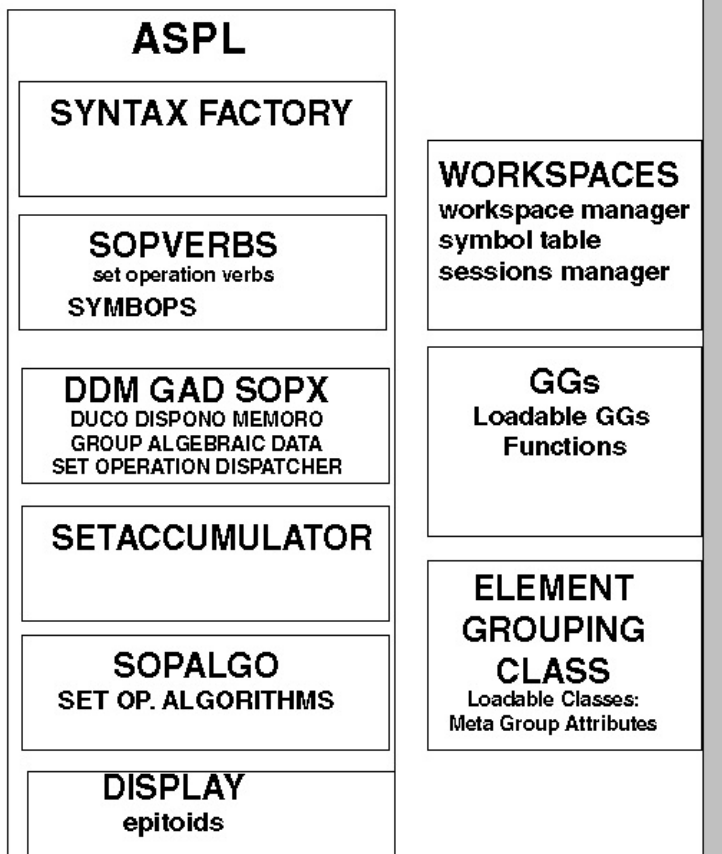
(\* footnote: An interpreter typically has its own virtual machine, but ASPL does not. The ASPL interpreter is built on top of pure Perl interpreter and it uses the Perl powerful virtual machine in executing its tasks.)

The TRIVIAL workspace is available with every ASPL distribution, and its grouping class is the ZEROGROUP which contains the bare information that is required by any grouping class in ASPL. The command **wid** displays the current workspace being loaded and its up time, and the command **egCwhoami** pings its grouping class container. Both commands are shown in Figure FFFFF.

**Note: ASPL Startup TRIVIAL ASPLv1.00-startup-trivial.png**

[full view](#)





-F- Fig. 1.1.4 [ASPL BLOCK][ASPL Internal Building Blocks]

ASPL © 2024 by Bassem W. Jamaledine

[Top Text]

-L- Listing. 1.1.1 [ASPLv1.00 Directory Tree ASPL-tree0.dir.numl][ASPLv1.00 Directory Tree]

(raw text)

```

1.      ASPLv1.00
2.      |-- bin
3.      |-- BRIDGE
4.      |   |-- ASPL
5.      |       |-- BAYLEVELGROUP
6.      |       |   |-- Feeder
7.      |       |   |-- EmStatv2
8.      |       |   |-- Cexec
9.      |       |   |-- EmVectors
10.     |       |   |-- Formattings
11.     |       |   |-- GGs
12.     |       |       |-- GE03TRI
13.     |       |       |   |-- CTXSETOP
14.     |       |       |   |-- OSCILLATORSGROUP
15.     |       |       |       |-- Feeder
16.     |       |       |   |-- SYSENVGROUP
17.     |       |       |       |-- Feeder
18.     |       |   |-- Groupings
19.     |       |   |-- Elements
20.     |       |       |-- ONEGROUP
21.     |       |       |   |-- ZEROGROUP
22.     |       |       |-- Helper
23.     |       |       |-- stubs
24.     |       |   |-- MockedGroupings
25.     |       |   |-- MockedObjects
26.     |   |-- bin
27. |-- etc
28. |-- lib
29.     |-- ASPL
30.         |-- ASPLSNTX
31.         |-- Directory
32.         |   |-- DDM
33.         |-- DISPLAY
34.         |-- MemUsage
35.         |-- PRIMITIVES
36.         |-- SETACCUMULATOR
37.         |-- SOPALGO
38.         |-- SOPVERBS
39.         |-- SOPX
40.         |-- SYMBOPS

```

```

41.         | | | `-- FTX_VERBPROCESSORS_BUILDER
42.         | | | `-- Utilities
43.         | | `-- Simple
44.         |-- license
45.         `-- shared
46.

```

ASPL © 2024 by Bassem Jamaledine

**[Top Text]**

**-L- Listing. 1.1.2 [ASPLv1.00 Directory Tree ASPL-tree0.dir.numl][ASPLv1.00 Directory Tree]**

[\(raw text\)](#)

```

1.          ASPLv1.00
2.          |-- bin
3.          |-- BRIDGE
4.          | |-- ASPL
5.          | | |-- BAYLEVELGROUP
6.          | | | `-- Feeder
7.          | | |-- EmStatv2
8.          | | | `-- Cexec
9.          | | |-- EmVectors
10.         | | |-- Formattings
11.         | | |-- GGs
12.         | | | |-- GE03TRI
13.         | | | | `-- CTXSETOP
14.         | | | |-- OSCILLATORSGROUP
15.         | | | | `-- Feeder
16.         | | | `-- SYSENVGROUP
17.         | | |   |-- Feeder
18.         | | |-- Groupings
19.         | | | |-- Elements
20.         | | | | |-- ONEGROUP
21.         | | | | `-- ZEROGROUP
22.         | | | |-- Helper
23.         | | | `-- stubs
24.         | | `-- MockedGroupings
25.         |   `-- MockedObjects
26.         `-- bin
27.     |-- etc
28.     |-- lib
29.         |-- ASPL
30.         | |-- ASPLSNTX
31.         | |-- Directory
32.         | | `-- DDM
33.         | |-- DISPLAY
34.         | |-- MemUsage
35.         | |-- PRIMITIVES
36.         | |-- SETACCUMULATOR
37.         | |-- SOPALGO
38.         | |-- SOPVERBS
39.         | |-- SOPX
40.         | |-- SYMBOPS
41.         | | `-- FTX_VERBPROCESSORS_BUILDER
42.         | `-- Utilities
43.         `-- Simple
44.     |-- license
45.     `-- shared
46.

```

ASPL © 2024 by Bassem Jamaledine

Starting ASPL in verbose mode will also reflect where ASPL loads its components at startup as shown in Figure FFFFF.

**ASPL Startup on the UNIX Shell Prompt Explained**

[full view](#)

```

File Edit View Bookmarks Settings Help
02:04 root@vienna: ~ # aspl WS1 -verbose

ASPL # 1.00

ASPL PROGRAM INSTALLED IN /opt/ASPLv1.00
ASPL WORKSPACE WILL IGNORE SESSIONS MANAGEMENT
ASPL WORKSPACE VARIABLES HAVE QUOTIENT-VARIABLES ENABLED

ASPL CLI # 1.00

PROGRAM DIRECTORY: /opt/ASPLv1.00
GROUPING CLASS NAME: POSIX
WORKSPACE REPOSITORY: /root/.aspl/WSP/WORKSPACES1
WORKSPACE NAME: WS1
LAST SAVED: 2023-02-21 02:37:07.655478
ELEMENT GROUPINGS CLASS: /opt/ASPLv1.00/BRIDGE/ASPL/Groupings
LOADABLE GGS DIRECTORY: /opt/ASPLv1.00/lib/ASPL/GGs
DDM CONFIGURATION: /opt/ASPLv1.00/etc/ddm.conf
DDM POOL: /root/.aspl/dev/DDM_DATA
MEMORY USED: 344408K

DISPLAY MODE: 2
LOGGING TO STDOUT
PERSIST VARIABLES IS OFF
RUN ASPL SCRIPT IN TWO PASSES
VARIABLES FREELY ASSIGNED

STARTING ASPL SESSION WITH WORKSPACE: WS1
NAMED GROUP: POSIX
last saved 2023-02-21 02:37:07.667119

aspl:1 2:4:28> 

```

workspace WS1 will be loaded from this repository

Grouping class POSIX defines the metadata attributes and loaded from this directory (not available with bridgeless)

GGs functions are loaded from this directory



**-F- Fig. 1.1.5 [ASPL Startup on the UNIX Shell Prompt Explained]**

ASPL © 2024 by Bassem Jamaledine

A user interacting with ASPL places a setadic operation to the interpreter. The interpreter submits it to the syntax factory, analyzes it, builds its object, and locates its verb. It then dispatches the object to its algorithmic set processing agent after fetching its symbols from either the answer stack or the symbol table. The result of the processed expression is finally saved in the symbol table, pushed on the top of the stack, and displayed on the terminal. The simplest way to see the flow of ASPL processing is to enable the tracing facility of ASPL (see Appendix CCCCC).

## ■ ASPL Commands with asplcmd

At this point we introduce the **asplcmd** command because it is a quick way to execute short ASPL statements through the UNIX shell prompt.

You can direct ASPL to execute some of its statements by issuing **asplcmd** at the shell prompt followed by a string. If the string is more than a word then it must be quoted since it is passed like a single argument to **asplcmd**. Without going into details on how **asplcmd** works, as this will become clear after reading the chapter "ASPL Scripts", it is used here to facilitate some explanation on ASPL displayable output.

The following command compares the datasets saved in the variable **mg123** of workspace **RANDONEBITMIX**.

```
# asplcmd 'load RANDONEBITMIX;gU mg123'
```

This command causes ASPL to load the workspace **RANDONEBITMIX**, then issues the **gU** to display the group unions found in the datasets saved in the variable **mg123**. The semicolon is used to separate the two statements. Observe the output showing the groups, subgroups, and elements. The next section explains the symbolic schemes and colors used by ASPL when displaying its output.

The following command compares two variables **a2** and **a7** that has been saved in workspace **WS1**.

```
# asplcmd 'load WS1; ks mtime chksum ppdd; gU a2 a7'
```

There are three statements separated by semicolons in this string passed to ASPL. The first statement load the workspace **WS1**, the second statement set the **ks** attribute vector to **mtime**



chksum ppdd, and the third statement display the group union of the datasets saved in a2 and a7.

When you pass a string to asplcmd the interpreter will parse the string and execute its contents then exit.

## ■ ASPL Symbolic Schemes and Colors

ASPL is not a GUI application and does not require any GUI API, it uses plain ASCII characters for its symbolic operators and identifiers, and uses the ANSI colors scheme provided by the UNIX terminal. (\* footnote: APL users should be alarmed as there is no APL Greek symbols in here!) At any time you can start ASPL at the UNIX shell prompt of a terminal (or even on a dumb terminal) or an X Window session. Figure FFFF-1 shows ASPL started in a KDE session.

Because the set comparison operators classify data into three basic categories: intersection, union, and difference, then it is recommended to display the output using three different colors. In addition, since a group may contain subgroups and elements, then some symbolic scheme is needed to show the group differences along these colors. ASPL uses both colors and symbolic schemes to facilitate the readability of its output. Using the ANSI colors that is readily available on all UNIX terminal, and compounding it with symbolic schemes will help to distinguish when subgroups and elements are unique, equal, or different.

Figure FFFF shows the partial result of the command discussed in the previous section:

```
# asplcmd 'load RANDONEBITMIX;gU mg123'
```

Recall that there is no restriction on the name of subgroups and elements, that is an element name can be the subgroup name in another group, and even can be the subgroup name in the same group (as long as it is at different level). This is similar to the UNIX filesystem where in the same directory you cannot have a directory and a file using the same name.

To understand the meaning of ASPL colors and symbolic schemes let's go over figure FFFF where the output has been labeled at five points as follow:

- ① **gU: f!)** this is the gU of g1 g2 g3 where a difference in the element has been detected: it is the subgroup df2 that was detected in the subgroup ./df2/df1
- ② **gU: +f)** element df3 only in subgroup ./df2/df1/ in group g3
- ③ **gU: d!)** subgroup df2 in subgroup ./df2/ in all groups g1 g2 and g3 but they differ
- ④ **gU: f==)** element df1 in subgroup ./df2/df2/ in all groups g1 g2 and g3 and is the same
- ⑤ **gU: g!)** mixed as df3 is ./df2/ is subgroup in groups g1 g2 and it is an element in group g3

Notice also the colors: equality shown in green, difference shown in red, and a loner shown in gray.

**Note: ASPL Symbolic Schemes and Colors g1g2g3-12345.png**

[full view](#)

<p>(1) Element df2 in subsubgroup ./df2/df1/ of g1 g2 and g3 but differ</p> <p>(2) Element df3 only in subsubgroup df2/df1/ in g3</p> <p>(3) subgroup ./df2/ in all three groups but differ</p> <p>(4) Element df1 in subsubgroup ./df2/df2/ of g2 g2 and g3 and being equal</p> <p>(5) Mixed df3 in subgroup ./df2/ of g1 g2 and g3; it is a subsubgroup in g1 and g2 while it is an element in g3</p>	<pre> gU: f!) ** CHANGED F) f1 changed pl g1 g2 g3 (&lt;./df2/df1/&gt; / df2) gU f-) df2 bit=1 chksum=1 entropy=1 ppdd=g1 ff1= /df2/df1/df2 aelm=mti gU f-) df2 bit=1 chksum=1 entropy=1 ppdd=g2 ff1= /df2/df1/df2 aelm=mti gU f-) df2 bit=0 chksum=0 entropy=1 ppdd=g3 ff1= /df2/df1/df2 aelm=mti gU: +f) ** ONLYIN F) f1 onlyd pl g3 (&lt;./df2/df1/&gt; / df3) gU: d!) ** CHANGED D) dir changed pl g1 g2 g3 (&lt;./df2/&gt; / df2) gU d-) df2 bit=0 chksum=0 entropy=1 ppdd=g1 ff1= /df2/df2 aelm=mti(1 gU d-) df2 bit=1 chksum=1 entropy=1 ppdd=g2 ff1= /df2/df2 aelm=mti(1 gU d-) df2 bit=0 chksum=0 entropy=1 ppdd=g3 ff1= /df2/df2 aelm=mti(1 gU: f==) ** EQUAL F) f1 equal pl g1 g2 g3 (&lt;./df2/df2/&gt; / df1) gU f=) df1 bit=1 chksum=1 entropy=1 ppdd=g1 ff1= /df2/df2/df1 aelm=mti gU f=) df1 bit=1 chksum=1 entropy=1 ppdd=g2 ff1= /df2/df2/df1 aelm=mti gU f=) df1 bit=1 chksum=1 entropy=1 ppdd=g3 ff1= /df2/df2/df1 aelm=mti gU: f!) ** CHANGED F) f1 changed pl g1 g2 g3 (&lt;./df2/df2/&gt; / df2) gU f-) df2 bit=0 chksum=0 entropy=1 ppdd=g1 ff1= /df2/df2/df2 aelm=mti gU f-) df2 bit=1 chksum=1 entropy=1 ppdd=g2 ff1= /df2/df2/df2 aelm=mti gU f-) df2 bit=0 chksum=0 entropy=1 ppdd=g3 ff1= /df2/df2/df2 aelm=mti gU: f==) ** EQUAL F) f1 equal pl g1 g2 g3 (&lt;./df2/df2/&gt; / df3) gU f=) df3 bit=0 chksum=0 entropy=1 ppdd=g1 ff1= /df2/df2/df3 aelm=mti gU f=) df3 bit=0 chksum=0 entropy=1 ppdd=g2 ff1= /df2/df2/df3 aelm=mti gU: g!) ** MIXED M) mixed both pl g1 g2 g3 (&lt;./df2/&gt; / df3) gU d-) df3 bit=0 chksum=0 entropy=1 ppdd=g1 ff1= /df2/df3 aelm=mti(1 gU d-) df3 bit=0 chksum=0 entropy=1 ppdd=g2 ff1= /df2/df3 aelm=mti(1 gU d-) df3 bit=0 chksum=0 entropy=1 ppdd=g3 ff1= /df2/df3 aelm=mti(1 gU: f==) ** EQUAL F) f1 equal pl g1 g2 (&lt;./df2/df3/&gt; / df1) gU f=) df1 bit=1 chksum=1 entropy=1 ppdd=g1 ff1= /df2/df3/df1 aelm=mti gU f=) df1 bit=1 chksum=1 entropy=1 ppdd=g2 ff1= /df2/df3/df1 aelm=mti gU: f==) ** EQUAL F) f1 equal pl g1 g2 (&lt;./df2/df3/&gt; / df2) gU f=) df2 bit=0 chksum=0 entropy=1 ppdd=g1 ff1= /df2/df3/df2 aelm=mti gU f=) df2 bit=0 chksum=0 entropy=1 ppdd=g2 ff1= /df2/df3/df2 aelm=mti gU: +f) ** ONLYIN F) f1 onlyd pl g1 (&lt;./df2/df3/&gt; / df3) gU: d!) ** CHANGED D) dir equal pl g2 g3 (&lt;./&gt; / df3) gU d=) df3 bit=1 chksum=1 entropy=1 ppdd=g1 ff1= /df2/df3/df3 aelm=mti gU d=) df3 bit=1 chksum=1 entropy=1 ppdd=g2 ff1= /df3 aelm=mti(17076 gU d=) df3 bit=1 chksum=1 entropy=1 ppdd=g3 ff1= /df3 aelm=mti(17076 </pre>
---	---

-F- Fig. 1.1.6 [SYMBOLIC Schemes AND Colors][ASPL Symbolic Schemes and Colors]

This example might be confusing, but it was selected on purpose to make it clear that the label names of subgroups and elements are immaterial to ASPL algorithmic routines.

Let's take a look at another practical example where a file name and a directory name might intersect on the UNIX filesystem. Assuming you have access to the /tmp directory, issue the following commands on your shell prompt:

```
# mkdir /tmp/foodir1
# mkdir /tmp/foodir1/abc
# mkdir /tmp/foodir2
# touch /tmp/foodir2/abc
# asplcmd "createworkspace TRANSIENT POSIX;ggdir(dir,/tmp/foodir1);ggdir(dir,/tmp/foodir2);gU"
```

The first two commands create a directory and a subdirectory /tmp/foodir1 and /tmp/foodir1/abc respectively, and the next two commands create a directory and a file /tmp/foodir2 and /tmp/foodir2/abc respectively. The last command call ASPL to show the difference between the two directories /tmp/foodir1 and /tmp/foodir2.

The statement *createworkspace TRANSIENT POSIX* tells ASPL to load the temporary TRANSIENT workspace with element grouping class POSIX. The next statement *ggdir(dir,/tmp/foodir1)* tells ASPL to call the grouping function *ggdir()* on directory /tmp/foodir1. These statements will become clearer in the next chapters. For now, if you have issued these command successfully then you are already using ASPL.

The result of comparing the comparing the directories is shown below: **Note: Example comparing a file and a subdirectory that have the same name foodir-commented.png**

[full view](#)

```
There are 2 sets representing: /tmp/foodir1 /tmp/foodir2
** gU:d!~) ** CHANGED D) dir changed  pL /tmp/foodir1 /tmp/foodir2 (<> / .)
gU d~) . uid=root gid=root mtime=2024-02-19 20:11:00 chksum=0 ppdd=/tmp/foodir1 ffl=
gU d~) . uid=root gid=root mtime=2024-02-19 20:11:57 chksum=6955650 ppdd=/tmp/foodir2 ffl=
** gU:g!~) ** MIXED M) mixed both  pL /tmp/foodir1 /tmp/foodir2 (<./> / abc)
subdirectory gU d~) abc uid=root gid=root mtime=2024-02-19 20:11:11 chksum=0 ppdd=/tmp/foodir1 ffl=./abc
and file by the gU f~) abc uid=root gid=root mtime=2024-02-19 20:11:57 chksum=4294967295 ppdd=/tmp/foodir2 ffl=./abc
same name

/tmp/foodir1 2 entries
/tmp/foodir2 2 entries

DONE PROCESSING gU
20:17 mm01: /tmp # []
```

**-F- Fig. 1.1.7 [Example comparing a file and a subdirectory that have the same name]**

ASPL © 2024 by Bassem W. Jamaledine

Appendix DDDD shows a summary of ASPL symbolic schemes and how the user can edit the color configuration settings.

## ■ Why do you need to use ASPL

There are myriad reasons why you need to use the ASPL interpreter: to detect and highlight changes of systemic data in a UNIX cloud environment, to validate configuration data in a UNIX cluster, to resolve collision of class names in JAR archives, to compare analogous PATH across systems, etc. The following figure shows the result of a UNIX system whose socket has changed states:

**Figure monitor-socket-state.png**

[full view](#)

```

File Edit View Bookmarks Settings Help
13:32:38 root@mm01 /root aspl:4 > ,fu sok12345 sok12345@1 sok12345@2 sok12345@3

There are 4 sets representing: 32036(sok12345) 32036(sok12345.2) 32036(sok12345.3) 32036(sok12345.1)

** fu: f1 -- CHANGED F) f1'changed pl 32036(sok12345) 32036(sok12345.2) 32036(sok12345.3) 32036(sok12345.1) (<./> / 9
40738501)
fu-) 940738501 sokloca=0.0.0.0 sokrema=1.0.0.18:74565 sokstate=TCP_LISTEN soktxq=00000000 sokrxq=00000001 sok
tract=00 sokuid=0 soktimo=0 sokinode=940738501
fu-) 940738501 sokloca=0.0.0.0 sokrema=1.0.0.18:74565 sokstate=TCP_LISTEN soktxq=00000000 sokrxq=00000000 sok
tract=00 sokuid=0 soktimo=0 sokinode=940738501
fu-) 940738501 sokloca=0.0.0.0 sokrema=1.0.0.18:74565 sokstate=TCP_LISTEN soktxq=00000000 sokrxq=00000000 sok
tract=00 sokuid=0 soktimo=0 sokinode=940738501
fu-) 940738501 sokloca=0.0.0.0 sokrema=1.0.0.18:74565 sokstate=TCP_LISTEN soktxq=00000000 sokrxq=00000000 sok
tract=00 sokuid=0 soktimo=0 sokinode=940738501
** fu: f1 -- CHANGED F) f1'changed pl 32036(sok12345) 32036(sok12345.2) 32036(sok12345.3) (<./> / 940738502)
fu-) 940738502 sokloca=1.0.0.18:266291 sokrema=1.0.0.18:74565 sokstate=TCP_CLOSE_WAIT soktxq=00000000 sokrxq=0
0000001 soktract=00 sokuid=0 soktimo=0 sokinode=940738502
fu-) 940738502 sokloca=1.0.0.18:266291 sokrema=1.0.0.18:74565 sokstate=TCP_ESTABLISHED soktxq=00000000 sokrxq=0
0000000 soktract=00 sokuid=0 soktimo=0 sokinode=940738502
fu-) 940738502 sokloca=1.0.0.18:266291 sokrema=1.0.0.18:74565 sokstate=TCP_CLOSE_WAIT soktxq=00000000 sokrxq=0
0000001 soktract=00 sokuid=0 soktimo=0 sokinode=940738502

32036(sok12345) 2 entries
32036(sok12345.2) 2 entries
32036(sok12345.3) 2 entries
32036(sok12345.1) 1 entries

DONE PROCESSING ,fu sok12345 sok12345@1 sok12345@2 sok12345@3

13:32:45 root@mm01 /root aspl:5 > []

```

-F- Fig. 1.1.8 [Figure Monitoring UNIX System Socket State]

ASPL © 2024 by Bassem Jamaledine

Furthermore one can use ASPL to do simulation of players throwing dice on a crap table. Figure FFFF shows the result of the simulation of players tossing dice: find the event when all the three players have the same outcome.

Figure randomdice.aspl-900.png

full view

```

File Edit View Bookmarks Settings Help
01:34 mm01 ~ # randomdice.aspl 900
SIMULATION FOR 3 PLAYERS THROWING 900 TIMES DICE ON A CRATABLE

DONE PROCESSING p1 = ggdice(player,player1,throws,900,die1trials, 5 $2,die2trials, 3 $3)
DONE PROCESSING p2 = ggdice(player,player2,throws,900,die1trials, 5 $2,die2trials, 3 $3)
DONE PROCESSING p3 = ggdice(player,player3,throws,900,die1trials, 5 $2,die2trials, 3 $3)

*****
# SHOW THE THROW NUMBERS WHEN ALL 3 PLAYERS HAVE ABSOLUTELY THE SAME OUTCOME
*****

There are 3 sets representing: player3 player2 player1

** fu: f1 -- EQUAL F) f1'equal pl player3 player2 player1 (<throw$73/> / dice)
fu-) dice faces=32 face1=3 face2=2 chksum=8 entropy=5.170 ppdd=player3 ffl=throw$73/dice aelm=player3(Bob)(32)(5)
fu-) dice faces=32 face1=3 face2=2 chksum=8 entropy=5.170 ppdd=player2 ffl=throw$73/dice aelm=player2(Steve)(32)(5)
fu-) dice faces=32 face1=3 face2=2 chksum=8 entropy=5.170 ppdd=player1 ffl=throw$73/dice aelm=player1(Dave)(32)(5)

player3 1 entries
player2 1 entries
player1 1 entries

DONE PROCESSING fu'ks= p1 p2 p3

*****
# SIMILARITY WHEN ALL 3 PLAYERS HAVE ABSOLUTELY THE SAME OUTCOME
*****

subset1 vs subset2      sim11v21
1  player3(3.p3) | player1(1.p1)  0.02556
2  player2(2.p2) | player1(1.p1)  0.03556
3  player2(2.p2) | player3(3.p3)  0.03444

player3(3.p3) | player1(1.p1) | ##### 0.02556
player2(2.p2) | player1(1.p1) | ##### 0.03556
player2(2.p2) | player3(3.p3) | ##### 0.03444

DONE PROCESSING sim'fflz p1 p2 p3

*****
# SHOW THE THROW NUMBERS WHEN ALL 3 PLAYERS HAVE THE SAME SUM
*****

There are 3 sets representing: player3 player2 player1

** fu: f1 -- EQUAL F) f1'equal pl player3 player2 player1 (<throw$73/> / dice)
fu-) dice faces=32 face1=3 face2=2 chksum=8 entropy=5.170 ppdd=player3 ffl=throw$73/dice aelm=player3(Bob)(32)(5)
fu-) dice faces=32 face1=3 face2=2 chksum=8 entropy=5.170 ppdd=player2 ffl=throw$73/dice aelm=player2(Steve)(32)(5)
fu-) dice faces=32 face1=3 face2=2 chksum=8 entropy=5.170 ppdd=player1 ffl=throw$73/dice aelm=player1(Dave)(32)(5)

** fu: f1 -- CHANGED F) f1'changed pl player3 player2 player1 (<throw$73/> / dice)
fu-) dice faces=31 face1=3 face2=1 chksum=3 entropy=5.170 ppdd=player3 ffl=throw$73/dice aelm=player3(Bob)(31)(4)
fu-) dice faces=31 face1=3 face2=1 chksum=3 entropy=5.170 ppdd=player2 ffl=throw$73/dice aelm=player2(Steve)(31)(4)
fu-) dice faces=31 face1=3 face2=1 chksum=3 entropy=5.170 ppdd=player1 ffl=throw$73/dice aelm=player1(Dave)(31)(4)

player3 2 entries
player2 2 entries
player1 2 entries

DONE PROCESSING fu'c= p1 p2 p3

*****
# SIMILARITY WHEN ALL 3 PLAYERS HAVE SAME SUM
*****

subset1 vs subset2      sim11v21
1  player3(3.p3) | player1(1.p1)  0.04667
2  player2(2.p2) | player1(1.p1)  0.06222
3  player2(2.p2) | player3(3.p3)  0.06333

player3(3.p3) | player1(1.p1) | ##### 0.04667
player2(2.p2) | player1(1.p1) | ##### 0.06222
player2(2.p2) | player3(3.p3) | ##### 0.06333

DONE PROCESSING sim'fflc p1 p2 p3

01:34 mm01 ~ #
01:34 mm01 ~ # []

```

-F- Fig. 1.1.9 [Figure Simulation for Three Players Throwing Dice]

ASPL © 2024 by Bassem Jamaledine

ASPL can also treat DNA sequences as datasets and you can toy with these sequences through its alignment algorithms. Figure FFFF shows a mutant when altering the sequence randomly, and figure FFFF show the DNA sequence alignment of two men.

Figure mutant.aspl-img1.png





whose attributes are tied to sensors monitoring the water level between two bays. Figure FFFF shows a simulation when the water level between the two bays is critical.



full view

```
File Edit View Bookmarks Settings Help
1:56:29 root@mm01 /root aspl:6 > ks mean123 mtme aelm
ASPL:6> ks mean123 mtme aelm
1:56:29 root@mm01 /root aspl:6 > ks mean123 mtme aelm

aStat = mean123 mtme aelm
bStat = mean123 mtme aelm

1:56:36 root@mm01 /root aspl:7 > fa'ks= CRITICAL BAY1 BAY2
ASPL:7> fa'ks= CRITICAL BAY1 BAY2
1:56:36 root@mm01 /root aspl:7 > fa'ks= CRITICAL BAY1 BAY2

There are 3 sets representing: CRITICAL BAY2 BAY1
** [6:7=] -- EQUAL F) f1=equal pl CRITICAL BAY1 (<WATERLEVEL/> / 1706770555)
fa: 1706770555 mean123=above7feet mtme=2024-02-01 01:55:55 aelm:time(1706770555)point1(2.5)point2(2.5)point3(2.5)mean123(above7feet)
fa: 1706770555 mean123=above7feet mtme=2024-02-01 01:55:55 aelm:time(1706770555)point1(8.47)point2(8.47)point3(8.47)mean123(above7feet)
fa: 1706770555 mean123=above7feet mtme=2024-02-01 01:55:55 aelm:time(1706770555)point1(4.39)point2(7.91)point3(7.91)mean123(above7feet)

CRITICAL 1 entries
BAY2 1 entries
BAY1 1 entries

DONE PROCESSING fa'ks= CRITICAL BAY1 BAY2

1:56:48 root@mm01 /root aspl:8 > fa'ks= WARN BAY1 BAY2
ASPL:8> fa'ks= WARN BAY1 BAY2
1:56:48 root@mm01 /root aspl:8 > fa'ks= WARN BAY1 BAY2

There are 3 sets representing: WARN BAY2 BAY1
** [6:7=] -- EQUAL F) f1=equal pl WARN BAY2 BAY1 (<WATERLEVEL/> / 1706770521)
fa: 1706770521 mean123=5-feet mtme=2024-02-01 01:55:21 aelm:time(1706770521)point1(2)point2(2)point3(2)mean123(5-feet)
fa: 1706770521 mean123=5-feet mtme=2024-02-01 01:55:21 aelm:time(1706770521)point1(9.29)point2(7.88)point3(2.14)mean123(5-feet)
fa: 1706770521 mean123=5-feet mtme=2024-02-01 01:55:21 aelm:time(1706770521)point1(9.29)point2(7.88)point3(2.14)mean123(5-feet)
** [6:7=] -- EQUAL F) f1=equal pl WARN BAY2 BAY1 (<WATERLEVEL/> / 1706770543)
fa: 1706770543 mean123=5-feet mtme=2024-02-01 01:55:43 aelm:time(1706770543)point1(2)point2(2)point3(2)mean123(5-feet)
fa: 1706770543 mean123=5-feet mtme=2024-02-01 01:55:43 aelm:time(1706770543)point1(9.72)point2(1.74)point3(3.89)mean123(5-feet)
fa: 1706770543 mean123=5-feet mtme=2024-02-01 01:55:43 aelm:time(1706770543)point1(4.43)point2(4.43)point3(6.07)mean123(5-feet)
** [6:7=] -- EQUAL F) f1=equal pl WARN BAY2 BAY1 (<WATERLEVEL/> / 1706770569)
fa: 1706770569 mean123=5-feet mtme=2024-02-01 01:56:09 aelm:time(1706770569)point1(2)point2(2)point3(2)mean123(5-feet)
fa: 1706770569 mean123=5-feet mtme=2024-02-01 01:56:09 aelm:time(1706770569)point1(4.13)point2(7.8)point3(9.83)mean123(5-feet)
fa: 1706770569 mean123=5-feet mtme=2024-02-01 01:56:09 aelm:time(1706770569)point1(4.03)point2(7.96)point3(5.93)mean123(5-feet)

WARN 3 entries
BAY2 3 entries
BAY1 3 entries

DONE PROCESSING fa'ks= WARN BAY1 BAY2

1:56:57 root@mm01 /root aspl:9 > fa'ks= CRITICAL BAY1
ASPL:9> fa'ks= CRITICAL BAY1
1:56:57 root@mm01 /root aspl:9 > fa'ks= CRITICAL BAY1

There are 2 sets representing: CRITICAL BAY1
** [6:7=] -- EQUAL F) f1=equal pl CRITICAL BAY1 (<WATERLEVEL/> / 1706770542)
fa: 1706770542 mean123=above7feet mtme=2024-02-01 01:55:42 aelm:time(1706770542)point1(2.5)point2(2.5)point3(2.5)mean123(above7feet)
fa: 1706770542 mean123=above7feet mtme=2024-02-01 01:55:42 aelm:time(1706770542)point1(4.9)point2(9.82)point3(5.99)mean123(above7feet)
** [6:7=] -- EQUAL F) f1=equal pl CRITICAL BAY1 (<WATERLEVEL/> / 1706770544)
fa: 1706770544 mean123=above7feet mtme=2024-02-01 01:55:44 aelm:time(1706770544)point1(2.5)point2(2.5)point3(2.5)mean123(above7feet)
fa: 1706770544 mean123=above7feet mtme=2024-02-01 01:55:44 aelm:time(1706770544)point1(9.62)point2(7.98)point3(8.47)mean123(above7feet)
** [6:7=] -- EQUAL F) f1=equal pl CRITICAL BAY1 (<WATERLEVEL/> / 1706770551)
fa: 1706770551 mean123=above7feet mtme=2024-02-01 01:55:51 aelm:time(1706770551)point1(2.5)point2(2.5)point3(2.5)mean123(above7feet)
fa: 1706770551 mean123=above7feet mtme=2024-02-01 01:55:51 aelm:time(1706770551)point1(2.75)point2(9.95)point3(8.61)mean123(above7feet)
** [6:7=] -- EQUAL F) f1=equal pl CRITICAL BAY1 (<WATERLEVEL/> / 1706770555)
fa: 1706770555 mean123=above7feet mtme=2024-02-01 01:55:55 aelm:time(1706770555)point1(2.5)point2(2.5)point3(2.5)mean123(above7feet)
fa: 1706770555 mean123=above7feet mtme=2024-02-01 01:55:55 aelm:time(1706770555)point1(8.75)point2(4.39)point3(7.91)mean123(above7feet)
** [6:7=] -- EQUAL F) f1=equal pl CRITICAL BAY1 (<WATERLEVEL/> / 1706770578)
fa: 1706770578 mean123=above7feet mtme=2024-02-01 01:56:18 aelm:time(1706770578)point1(2.5)point2(2.5)point3(2.5)mean123(above7feet)
fa: 1706770578 mean123=above7feet mtme=2024-02-01 01:56:18 aelm:time(1706770578)point1(9.29)point2(7.59)point3(2.98)mean123(above7feet)

CRITICAL 5 entries
BAY1 5 entries

DONE PROCESSING fa'ks= CRITICAL BAY1

1:57:14 root@mm01 /root aspl:10 > fa'ks= CRITICAL BAY2
ASPL:10> fa'ks= CRITICAL BAY2
1:57:14 root@mm01 /root aspl:10 > fa'ks= CRITICAL BAY2

There are 2 sets representing: CRITICAL BAY2
** [6:7=] -- EQUAL F) f1=equal pl CRITICAL BAY2 (<WATERLEVEL/> / 1706770535)
fa: 1706770535 mean123=above7feet mtme=2024-02-01 01:55:35 aelm:time(1706770535)point1(2.5)point2(2.5)point3(2.5)mean123(above7feet)
fa: 1706770535 mean123=above7feet mtme=2024-02-01 01:55:35 aelm:time(1706770535)point1(5.93)point2(8.87)point3(7.3)mean123(above7feet)
** [6:7=] -- EQUAL F) f1=equal pl CRITICAL BAY2 (<WATERLEVEL/> / 1706770538)
fa: 1706770538 mean123=above7feet mtme=2024-02-01 01:55:38 aelm:time(1706770538)point1(2.5)point2(2.5)point3(2.5)mean123(above7feet)
fa: 1706770538 mean123=above7feet mtme=2024-02-01 01:55:38 aelm:time(1706770538)point1(9.43)point2(7.82)point3(4.11)mean123(above7feet)
** [6:7=] -- EQUAL F) f1=equal pl CRITICAL BAY2 (<WATERLEVEL/> / 1706770549)
fa: 1706770549 mean123=above7feet mtme=2024-02-01 01:55:49 aelm:time(1706770549)point1(2.5)point2(2.5)point3(2.5)mean123(above7feet)
fa: 1706770549 mean123=above7feet mtme=2024-02-01 01:55:49 aelm:time(1706770549)point1(6.24)point2(9.15)point3(8.13)mean123(above7feet)

root: aspl root: aspl root: aspl ASPL: aspl shared: bash
```

-F- Fig. 1.1.12 [Figure Monitoring the Water Level Between Two Bays]

ASPL © 2024 by Bassem Jamaledine

ASPL can do fuzzy set operations, these are simple operators like **y&**, **yU**, and **y\** to get the fuzzy intersection, union, and difference respectively.

With its powerful regular expression processor, ASPL can do shallow set operations. Just type **shallowed** and you select the shallow matching that you desire.

A quick view of the shallow table module is shown below.

```
aspl:1 > shallowed
```

THE SHALLOW MATCHING IDENTIFIER TO SELECT THE ROUTINE WHEN SHALLOW SETOPS ARE USED:

IDENTIFIER	DESCRIPTION
nothing	matching nothing at all
starstar	matching anything and everything
matchandmatch	matching the ./subgroups and the element
matchormatch	matching the ./subgroups or the element
elem	matching just the element and ignoring ./subgroups
stem	matching just the ./subgroups and ignoring the element
endjoinedeither	matching from end of ./subgroups/element for either
begjoinedeither	matching the beginning of ./subgroups/element for either
endstem	matching just the ./subgroups from the end and ignoring the element
endstemeither	matching just the ./subgroups from the end and ignoring the element (for either)
begstem	matching from beginning of ./subgroups while ignoring the element
begstemeither	matching from beginning of ./subgroups while ignoring the element (for either)
piecedstem	matching at least one piece in ./subgroups while ignoring the element
piecedstemelem	matching the element and at least one piece in the ./subgroups
begelem	matching from the beginning of element and ignoring ./subgroups (for either)
endelem	matching from the end of element and ignoring ./subgroups (for either)
rgxelem	matching the element anywhere and ignoring ./subgroups (for either)
precrelem	apply processor when matching the element anywhere and ignoring ./subgroups (for ei

CURRENTLY LOADED piecedstemelem

WHEN SELECTING prcrelem AS THE SHALLOW MATCHING IDENTIFIER, ONE OF THE FOLLOWING NODE PROCESSOR IDENTIFIER CAN BE SELECTED:

IDENTIFIER	EVAL	DESCRIPTION
transac	1	capture the element where word Transaction occurred, ignore case
cla2ja	1	substitute .class with .java
ja2cla	1	substitute .java with .class
uc	1	upper case
lc	1	lower case
asis	1	neutral without any change

CURRENTLY LOADED PROCESSOR asis

THE shallowedMatches PACKAGE CAN BE EDITED TO ADD MORE MATCHING SUBROUTINES. SEE ASPL CONFIGURATION FILES FOR MORE ABOUT EDITING shallowedMatches PACKAGE.

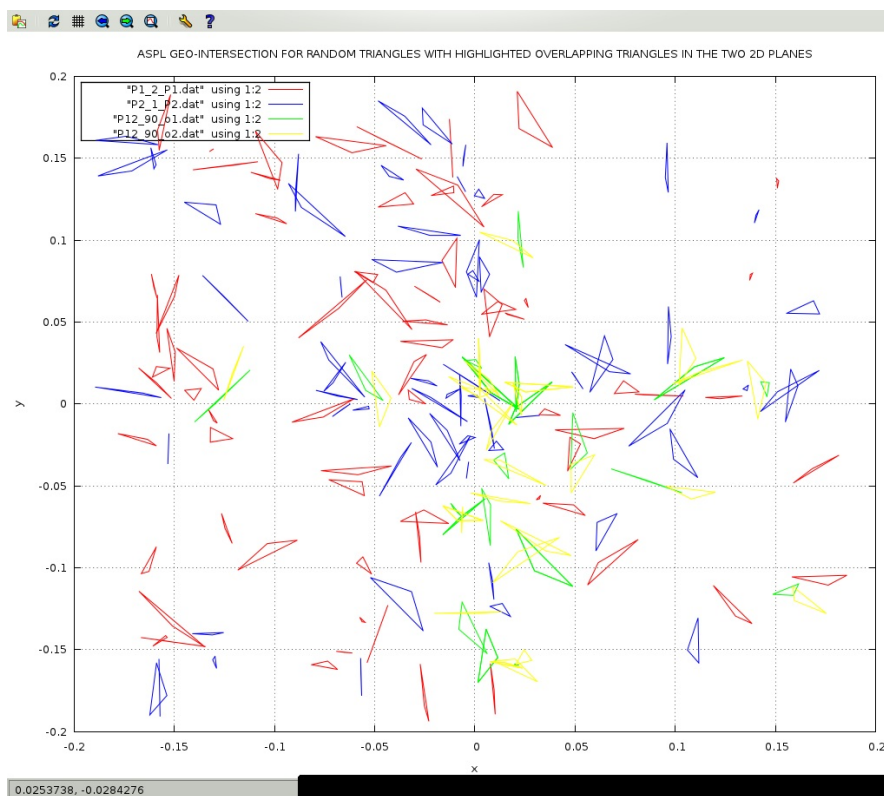
Moreover ASPL can switch its set operation mode to do geometric set operations so that you can detect intersecting polygons in 2D planes or on spheres.

Figure FFFFF shows the intersecting polygons in two planes. The intersecting polygons are contrasted by showing their vertex in squares.

P12\_90-both-and-intersect-img2.png

**Figure P12\_90-both-and-intersect-img2.png**

[full view](#)



**-F- Fig. 1.1.13 [Figure ASPL GEO-INTERSECTION FOR RANDOM TRIANGLES WITH HIGHLIGHTED OVERLAPPING TRIANGLES IN THE TWO 2D PLANES]**

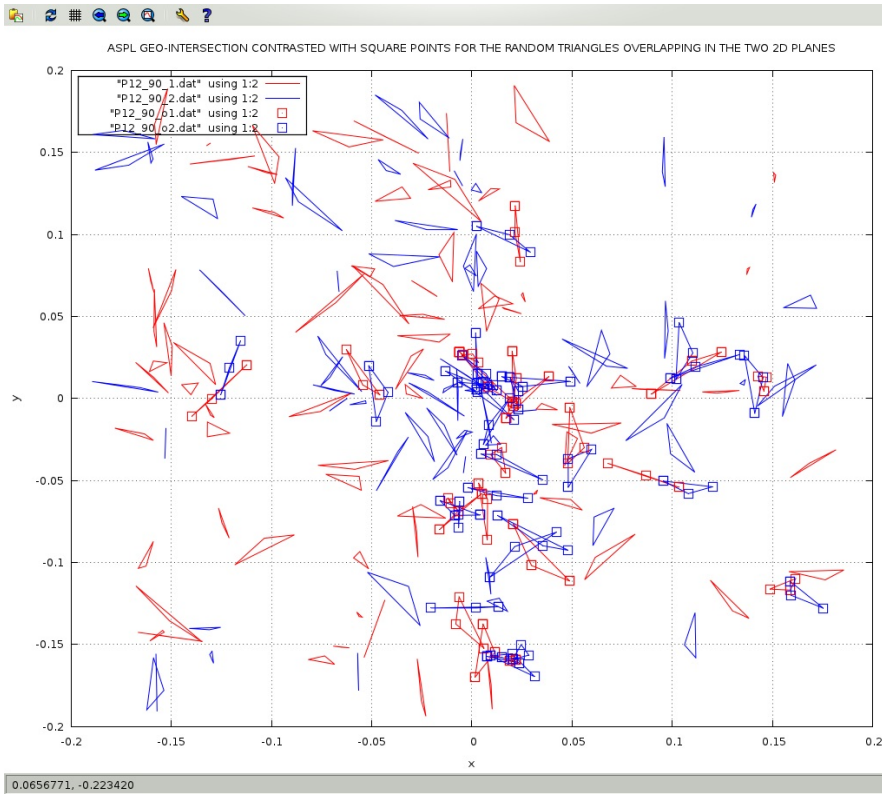
ASPL © 2024 by Bassem Jamaledine

P12\_90-both-and-intersect-img1.png

PUBLISHER VERSION USING SQUARE POINTS FOR BLACK AND WHITE PRINT

**Figure P12\_90-both-and-intersect-img1.png**

[full view](#)



**-F- Fig. 1.1.14 [Figure ASPL GEO-INTERSECTION CONTRASTED WITH SQUARE POINTS FOR THE RANDOM TRIANGLES OVERLAPPING IN THE TWO 2D PLANES]**

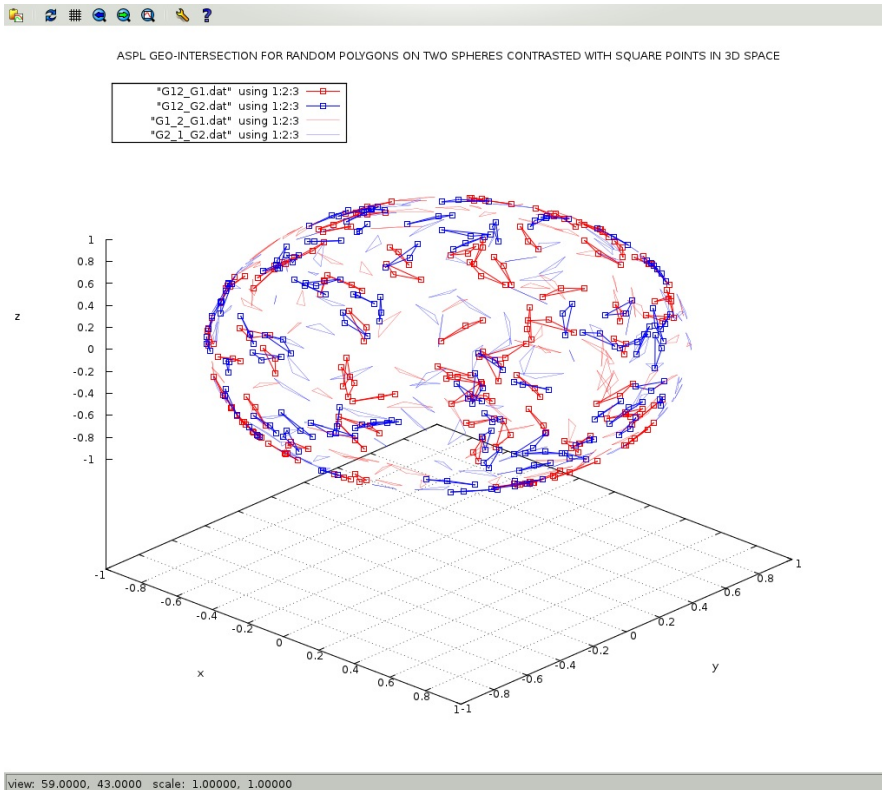
ASPL © 2024 by Bassem Jamaledine

Figure FFFFF shows the intersecting polygons on two spheres, and figure FFFF shows the intersects and differences between these polygons.

G12\_170-all-with-intersect-img3D.png

**Figure G12\_170-all-with-intersect-img3D.png**

[full view](#)



**-F- Fig. 1.1.15 [Figure RANDOM POLYGONS OVERLAPPING ON TWO SPHERES IN 3D SPACE]**

ASPL © 2024 by Bassem Jamaledine



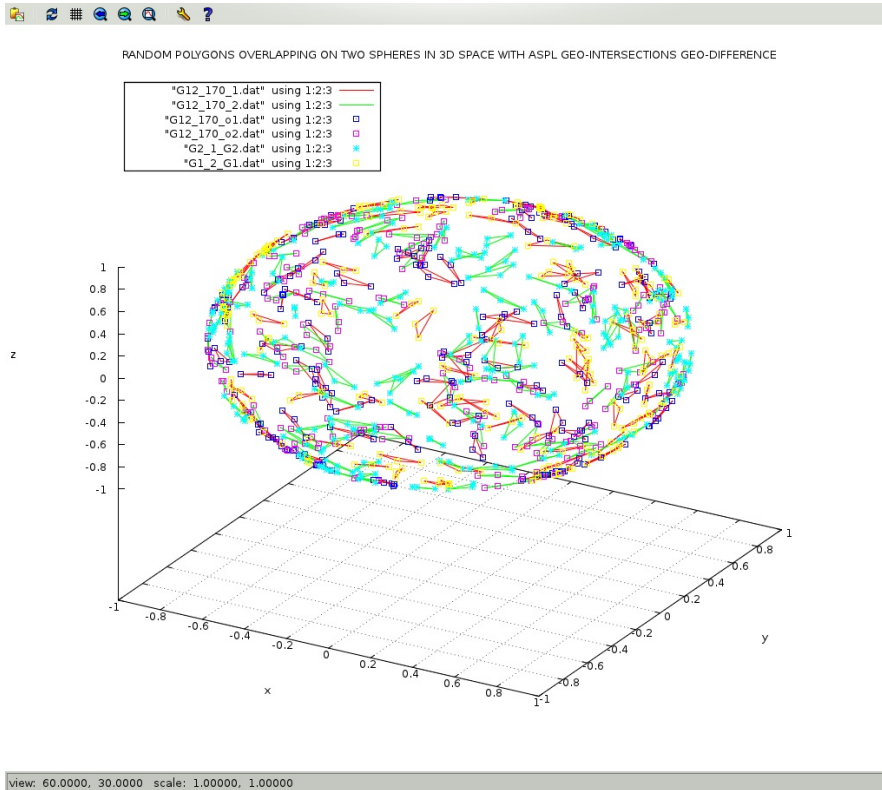
Refer to **WARN: IN Introduction-to-ASPL.raw THE FOLLOWING Ref: CANNOT BE RESOLVED**

**<Ref:-see section -chaptit "RANDOM POLYGONS ON THREE SPHERES" -ptt "RANDOM POLYGONS ON THREE SPHERES" />**

G12\_170-both-all-img3D.png

**Figure G12\_170-both-all-img3D.png**

[full view](#)



**-F- Fig. 1.1.16 [Figure RANDOM POLYGONS OVERLAPPING ON TWO SPHERES IN 3D SPACE WITH ASPL GEO-INTERSECTIONS GEO-DIFFERENCE]**

ASPL © 2024 by Bassem Jamaledine

Many of these examples are detailed in **WARN: IN Introduction-to-ASPL.raw THE FOLLOWING Ref: CANNOT BE RESOLVED**

**<Ref:-see section -chaptit "ASPL by Examples" -ptt "ASPL by Examples" />**

Chapter Ref:ASPL by Examples

In addition the interpreter offers *the ASPL scripting language* so that you can enhance your system with powerful commands. Consider the following script that shows the differences in JAR archives. It can be called with a simple command `jarcompare.aspl jarfile1 jarfile2`.

**[Top Text]**

**LISTING jarcompare.aspl ASPL Script jarcompare.aspl**

[\(raw text\)](#)

```
1.      #!/usr/bin/env aspl
2.      #ENVARG= -groupingclass POSIX -wsname TRANSIENT -singlepass
3.
4.      ;,*****
5.      ;; jarcompare.aspl
6.      ;; Compare two JAR archives
7.      ;; Copyright © 2024 Bassem W. Jamaledine
8.      ;;
9.      ;,*****
10.
11.     endScriptIfShellArgsLessThan 2
12.
13.     ;;DEF FN cmp2sets := {gU {g\, %1 %2}{g\, %2 %1}{g&, %1 %2}}
14.
15.     timeout 60
16.     displayoff
17.     d1 = ggjar(jarfile,$1,calchksm,1,calentropy,1)
18.     d2 = ggjar(jarfile,$2,calchksm,1,calentropy,1)
19.
20.     displayon
21.     ks chksm size ffl
```



```

22. ;;print #####
23. ;;print # SHOWING SET COMPARISONS BETWEEN $1 and $2
24. ;;print #####
25. ;;FN cmp2sets(d1,d2)
26. print #####
27. print # SHOWING SET INTERSECTION WITH DIFFERENT CHECKSUMS BETWEEN $1 and $2
28. print #####
29. f&`c~ d1 d2
30. print #####
31. print # SHOWING SET SIMILARITY BETWEEN $1 and $2
32. print #####
33. sim d1 d2
34. println
35.
36. endscrip
37.
38. _END_
39.
40. $00 compares two JAR archives
41.
42. $00 must be followed by the names of two JAR archives
43.
44. Example:
45. To compare the two JAR archives /tmp/TX/27238-tx.jar and
   /tmp/TX/38141-tx.jar
46. $00 /tmp/TX/27238-tx.jar /tmp/TX/38141-tx.jar
47.

```

ASPL © 2024 by Bassem Jamaledine

Figure FFFFF shows the result of jarcompare.aspl when comparing two JAR archives. Notice how the colliding names of the classes have been detected and their different checksums is displayed in red color.

**Figure jarcompare.aspl-img.png**

[full view](#)

```

File Edit View Bookmarks Settings Help
00:58 mm01: /opt/ASPLv1.00/shared # jarcompare.aspl /tmp/TX/38478-tx.jar /tmp/TX/45240-tx.jar
TIMEOUT IS SET TO 60 SECONDS
DONE PROCESSING d1 = ggjar(jarfile,/tmp/TX/38478-tx.jar,calchksm,1,calentropy,1)
DONE PROCESSING d2 = ggjar(jarfile,/tmp/TX/45240-tx.jar,calchksm,1,calentropy,1)

aStat = chksm size ff1
bStat = chksm size ff1

#####
# SHOWING SET INTERSECTION WITH DIFFERENT CHECKSUMS BETWEEN /tmp/TX/38478-tx.jar and /tmp/TX/45240-tx.jar
#####
There are 2 sets representing: /tmp/TX/38478-tx.jar /tmp/TX/45240-tx.jar
** f6: [1] -- CHANGED F1 f1 changed pl /tmp/TX/38478-tx.jar /tmp/TX/45240-tx.jar (<./META-INF/> / MANIFEST.MF)
f6- MANIFEST.MF chksm=1228424340 size=207 ff1s /META-INF/MANIFEST.MF
f6- MANIFEST.MF chksm=2228958119 size=210 ff1s /META-INF/MANIFEST.MF
** f6: [1] -- CHANGED F1 f1 changed pl /tmp/TX/38478-tx.jar /tmp/TX/45240-tx.jar (<./META-INF/> / ws-server-startup.xml)
f6- ws-server-startup.xml chksm=3036993950 size=270 ff1s /META-INF/ws-server-startup.xml
f6- ws-server-startup.xml chksm=305447080 size=253 ff1s /META-INF/ws-server-startup.xml
** f6: [1] -- CHANGED F1 f1 changed pl /tmp/TX/38478-tx.jar /tmp/TX/45240-tx.jar (<./com.ibm.ws.LocalTransaction/> / LocalTransactionCoordinator.class)
f6- LocalTransactionCoordinator.class chksm=1141020310 size=901 ff1s /com.ibm.ws.LocalTransaction/LocalTransactionCoordinator.class
f6- LocalTransactionCoordinator.class chksm=1087872554 size=925 ff1s /com.ibm.ws.LocalTransaction/LocalTransactionCoordinator.class
** f6: [1] -- CHANGED F1 f1 changed pl /tmp/TX/38478-tx.jar /tmp/TX/45240-tx.jar (<./com.ibm.ws.LocalTransaction/> / LocalTransactionCurrent.class)
f6- LocalTransactionCurrent.class chksm=2678772339 size=801 ff1s /com.ibm.ws.LocalTransaction/LocalTransactionCurrent.class
f6- LocalTransactionCurrent.class chksm=142220068 size=830 ff1s /com.ibm.ws.LocalTransaction/LocalTransactionCurrent.class
** f6: [1] -- CHANGED F1 f1 changed pl /tmp/TX/38478-tx.jar /tmp/TX/45240-tx.jar (<./com.ibm.ws.Transaction/> / TxProperties.class)
f6- TxProperties.class chksm=780699889 size=2255 ff1s /com.ibm.ws.Transaction/TxProperties.class
f6- TxProperties.class chksm=385153173 size=2297 ff1s /com.ibm.ws.Transaction/TxProperties.class
** f6: [1] -- CHANGED F1 f1 changed pl /tmp/TX/38478-tx.jar /tmp/TX/45240-tx.jar (<./com.ibm.ws.uow/> / UOWManager.class)
f6- UOWManager.class chksm=1810932839 size=496 ff1s /com.ibm.ws.uow/UOWManager.class
f6- UOWManager.class chksm=389208897 size=497 ff1s /com.ibm.ws.uow/UOWManager.class

/tmp/TX/38478-tx.jar 6 entries
/tmp/TX/45240-tx.jar 6 entries
DONE PROCESSING f6`c~ d1 d2

#####
# SHOWING SET SIMILARITY BETWEEN /tmp/TX/38478-tx.jar and /tmp/TX/45240-tx.jar
#####
Similarity: Similarity
/tmp/TX/45240-tx.jar (G:G) | /tmp/TX/38478-tx.jar(1:G1) 0.91937

DONE PROCESSING sim d1 d2
01:04 mm01: /opt/ASPLv1.00/shared # []

```

**-F- Fig. 1.1.17 [Figure Comparing Two JAR Archives]**

ASPL © 2024 by Bassem Jamaledine

Naturally the ASPL interpreter allows you to do set operation interactively at its prompt and the user can repeat the same operations shown in the script at the ASPL prompt. Writing scripts with ASPL is simple as you will see in Chapter "ASPL Scripts".

ASPL uses the notion of grouping class and treats algebraic groups per their meta data. The grouping class specially characterizes the set of data being treated.

ASPL Elements Class Containment is discussed in Chapter "ASPL A DETAILED VIEW".

ASPL is premier set calculator uniquely identified with its powerful programming of set elements and their attributes: ASPL element attributes can be statically defined, or dynamically updatable by anonymous subroutines, or refreshable with new data when tied to generative devices. When a change is detected in an element, ASPL archives the variable, and you can use set operators to display what has changed.

