ASIStint: An Interactive ASIS Interpreter

Vasiliy Fofanov
fofanov@bigfoot.com

Sergey Rybin
rybin@possum.srcc.msu.su

Alfred Strohmeier
alfred.strohmeier@di.epfl.ch

Contents

- Introduction
  - What is ASIS
  - What is ASIStint
- Example of ASIStint usage
- ASIStint language
- The ASIStint execution environment
- Uses and benefits of ASIStint
  - Learning ASIS and experimenting with it
  - Testing an ASIS implementation
- References on the WWW

What is ASIS

ASIS = **A**da **S**emantic **I**nterface **S**pecification

- an interface between an Ada environment and any tool or application, called an ASIS application, requiring information from this environment
- a set of self-documented Ada package specifications:
  - types
  - operations (or ASIS queries)
- a conventional ASIS application **with**s/**use**s the necessary ASIS packages to gain access to the corresponding ASIS functionality

ASIS is currently undergoing ISO standardization process

What is ASIStint

ASIStint = **ASIS t**[est] **int**[erpreter]
(pronounced like ***assistant***)

- a regular ASIS application
- allows to call ASIS queries interactively
- scripting facilities
- session logging

Uses of ASIStint:

- learning ASIS
- experimenting with ASIS queries
- debugging and testing of ASIS implementations

Current state:

Available as part of the public distribution of ASIS-for-GNAT

Example of ASIStint usage

An Ada Unit

```
with Text_IO;
procedure Demo is
   A : Integer := 1;
   B : Integer := 2;
   C : Integer;
begin
   C := A + B;
   Text_IO.Put_Line (Integer'Image(C));
end Demo;
```

A Problem

Output the first name declared immediately within this unit, i.e. "**A**".

Example of ASIStint usage

An ASIS solution

```
with Text_IO;
with Asis;
with Asis.Implementation;
use Asis.Implementation;
...
<with and use all the other required ASIS
packages>
...
procedure Demo_ASIS is
    Ctx   : Asis.Context;
    Unit  : Asis.Compilation_Unit;
    Decl  : Asis.Element;
begin -- Demo_ASIS
    Initialize("");
    Associate(Ctx, "", "");
    Open(Ctx);
    Unit := Compilation_Unit_Body("Demo", Ctx);
    Decl := Unit_Declaration(Unit);
```

Example of ASIStint usage

An ASIS solution (cont.)

```
declare -- block
   Body_Decl : Asis.Element_List :=
      Body_Declarative_Items(Decl, True);
   -- We need a block statement here
   Decl_Names : Asis.Element_List :=
      Names(Body_Decl(1));
begin -- block
   Text_IO.Put_Line(
      Defining_Name_Image(Decl_Names(1)));
end; -- block
end Demo_ASIS;
```

Example of ASIStint usage

An ASIStint solution

set(Ctx)

Initialize("")

Associate(Ctx, "", "")

Open(Ctx)

set(Unit, Compilation_Unit_Body ("Demo", Ctx))

set(Decl, Unit_Declaration(Unit))

set(Body_Decl, Body_Declarative_Items (Decl, True))

-- <u>**Notice the difference with the Ada code:**</u>

-- <u>**we don't need a block statement!**</u>

set(Decl_Names, Names(Body_Decl(1)))

print(Defining_Name_Image (Decl_Names(1)))

ASIStint language

The design goals

- immediate interpretation
- interactive use

Functional language

- an ASIStint program is a sequence of function calls
- function calls may be nested
- an ASIStint function call follows the Ada syntax for a function call

ASIStint functions classification

- ASIS queries
- ASIStint-specific functions:
  - ASIStint utilities
  - ASIStint service functions

ASIS 95 support

all ASIS queries, except ***Traverse_Element***

ASIStint language

Data Types

- support for the predefined Ada types
  - **Integer** (operations: ***Add***, ***Sub***, ***Eq***, ***Lt***, and ***Gt***)
  - **Boolean** (operations: ***Not***, ***And***, and ***Or***)
  - **String** (operations: ***Concat***, ***Eq***, ***Lt***, and ***Gt***)
  - literals for types Integer, Boolean and String are supported
- support for all ASIS composite types
- the values of ASIS enumeration types are mapped to string literals, in conformance to the **‘Image** attribute
- support for ASIS lists (*Element_List*, etc.)
  - indexing is defined
  - the function ***Length*** yields the length of a list

ASIStint language

Variables

- ASIStint variables are used to store values returned from queries, in order to display them, and in order to pass them to subsequent calls
- ASIStint variables can be of any ASIStint type
- a variable name is an Ada-style identifier
- variables are dynamically created and assigned by the ASIStint ***Set*** utility
- no predefined variables at the start of a session

ASIStint language

ASIStint utilities

- **Help system:**

  Help [(<query-name>)]

  Info

- **Environment control:**

  Quit [(<exit-status>)]

  Log [("filename")]

  Loglevel (<level>)

  Run [("filename")]

  Pause

  Exit

- **Variable handling:**

  Set (ID [, <expression>])

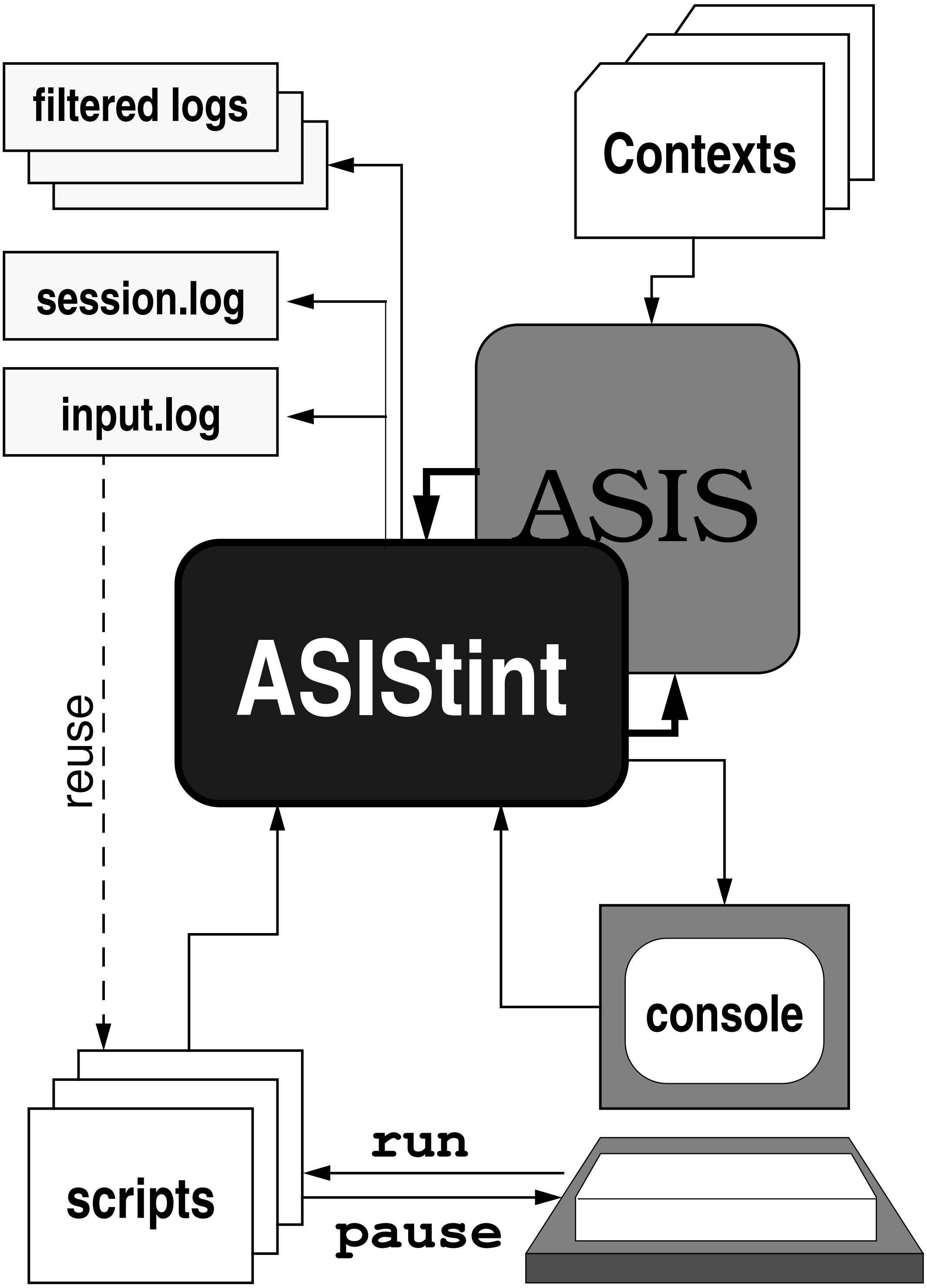
- **Output:**

  Print (<expression>)

- **Conditional execution:**

  If (<boolean_expression>,
  <ASIStint_command1>
  [, <ASIStint_command2>])

ASIStint execution environment



Uses and benefits of ASIStint

Learning ASIS and experimenting with it

To get acquainted with ASIS, it is helpful to be able to play around with ASIS queries, to see how they should be used, and what information they return

ASIStint is an interactive interpreter

- "on-line" experimenting with ASIS - the user sees immediately if what he gets is what he wants
- "error-forgiving" capability - user errors do not lead to the termination of the interactive session, since all exceptions raised by ASIS are intercepted
- on-line help on ASIS queries syntax

Uses and benefits of ASIStint

Debugging and testing ASIS implementations

Testing a program means executing it to find errors in its implementation.

test set = input data + expected results

ASIS - no "main" program =>
to apply the approach, many "main" programs have to be written.

Disadvantages:

extremely cumbersome, time-consuming and resource-intensive: average test will contain

- an Ada program (this is the data part of the test run)
- an ASIS application (also an Ada program)

both must be compiled, but may contain errors and not pass the first compilation

Uses and benefits of ASIStint

Debugging and testing ASIS implementations: ASIStint approach

- interactive input of test cases => instant response from ASIStint
- the input and results of such an interactive session are recorded
- the corresponding script can be replayed, and the new results compared with the old one (e.g. for regression testing)

ASIS-for-GNAT experience

ASIStint was used for debugging and regress testing of ASIS-for-GNAT implementation.

References on the WWW

ASIS documents are available electronically
on the World Wide Web:

**http://www.acm.org/sigada/WG/
asiswg/asiswg.html**

or by anonymous ftp:

**sw-eng.falls-church.va.us/public/
AdaIC/work-grp/asiswg**

ASIS-for-GNAT is available electronically
from ASIS-for-GNAT Home Page:

http://such.srcc.msu.su/asis/

or from LGL-EPFL by anonymous ftp:

ftp://lglftp.epfl.ch/pub/ASIS