Babel Assertion and Method Hook Basics

PROTOTYPE

CCA Forum
Portland, Oregon
April 28, 2005

Tamara L. Dahlgren
dahlgren1@llnl.gov

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

UCRL-PRES-xxxxxx

Babel Assertion and Method Hook Basics

- ➡ Assertion support
  - Babel options
  - SIDL grammar
  - Assertion functions
  - Assertion violation actions
  - Enforcement options
- Pre/Post method hooks
  - Babel options
  - Generated server
  - Client code

Only relevant for assertion-annotated SIDL files!

Enforcement generation can be disabled at compile-time.

- Enabled by default!
- Equivalent option

```
$ babel --server=c     \
   --repository-path=$(REPO) \
   --assertion-level=1 vector.sidl
```

- Disabled by setting level to 0

```
$ babel –sc -R$(REPO) -a0 vector.sidl
```

vector.sidl example from regression tests.

```
package vector version 1.0 {
  class Utils { …

    static double norm(in array<double> u, in double tol, in int badLevel)
      throws
        sidl.PreconditionViolation, NegativeValueException,
        sidl.PostconditionViolation;                          Implicit!
      require
        not_null : u != null;
        u_is_1d : dimen(u) == 1;                               Preconditions
        non_negative_tolerance : tol >= 0.0;
      ensure
        no_side_effects : is pure;
        non_negative_results : result >= 0.0;                  Postconditions
        nearEqual(result, 0.0, tol) iff isZero(u, tol);
  … }
}
```

vector.sidl

Operators, method calls, and literals are supported.

Operators	Logical	iff[1], implies[1], or, xor, and[1]
	Relational	==[1], !=[1], <, <=, >=[1], >[1]
	Additive	+[1], -
	Multiplicative	**, *, /, mod, rem, <<, >>
	Unary	not, is[1]
Logical grouping		()[1]
Method calls		<name> '(' [<argument-list>] ')' [1]
Terminals		<boolean>, <char>[2], <dcomplex>[2], <double>, <enumerator>[1], <fcomplex>[2], <float>[1], <identifier>[1], <integer>[1], <long>, <string>[2]
Literal keywords		true, false, null[1], result[1], pure[1]

[1]Exercised so far.
[2]Implementation incomplete.

norm() uses two built-in and one user-defined assertion function.

```
package vector version 1.0 {
  class Utils { …

    static double norm(in array<double> u, in double tol, in int badLevel)
      throws
        sidl.PreconditionViolation, NegativeValueException,
        sidl.PostconditionViolation;

      require
        not_null : u != null;
        u_is_1d : dimen(u) == 1;                          } Preconditions
        non_negative_tolerance : tol >= 0.0;

      ensure
        no_side_effects : is pure;
        non_negative_results : result >= 0.0;             } Postconditions
        nearEqual(result, 0.0, tol) iff isZero(u, tol);
  … }                                      Utils.isZero(u, tol)
}
```

vector.sidl

Built-in assertion functions provide more expressiveness.

O(1)-time		O(n)-time		
dimen[4]	range	all[3]	max	nonIncr
irange	size[4]	any[3]	min	none[3]
lower	stride	count[3]	nearEqual[4]	range
nearEqual	upper	irange[4]	nonDecr[4]	sum

[3]Array parameters can appear on lhs and/or rhs of relational expression.
[4]Exercised so far.

Babel supports eight built-in O(1) time assertion functions.

Function	Returns…
dimen(u)	Dimension of array u.
irange(x, n_{low}, n_{high})	True for x, n_{low}, n_{high} in integers, if $n_{low} \leq x \leq n_{high}$.
lower(u, n)	Lower index of the n^{th} dimension of array u.
nearEqual(x, y, t)	True for x, y, and tolerance t in reals, $t \geq 0.0$, if $\lvert x \rvert - \lvert y \rvert \leq t$.
range(x, r_{low}, r_{high}, t)	True for x, r_{low}, r_{high} and tolerance t in reals, $t \geq 0.0$, if $r_{low} - t \leq x \leq r_{high} + t$.
size(u)	Allocated size of array u.
stride(u, n)	Stride of the n^{th} dimension of array u.
upper(u, n)	Upper index of the n^{th} dimension of array u.

Handout Slide

Also eight basic built-in O(n) assertion functions for arrays.

Function	Returns…
irange(u, n_{low}, n_{high})	True for n_{low}, n_{high} in integers, if $\forall$ u_i in integer array u, $n_{low} \leq u_i \leq n_{high}$.
max(u)	u_m in array u such that $u_m \geq u_i$ $\forall$ i.
min(u)	u_m in array u such that $u_m \leq u_i$ $\forall$ i.
nearEqual(u, v, t)	True if tolerance t in reals, $t \geq 0.0$, $\forall$ u_i in real array u and v_j in real array v, $\vert u_i \vert - \vert v_j \vert \leq t$.
nonDecr(u)	True if $\forall$ u_i, u_j in array u, $i < j$, $u_i \leq u_j$.
nonIncr(u)	True if $\forall$ u_i, u_j in array u, $i < j$, $u_i \geq u_j$.
range(u, r_{low}, r_{high}, t)	True for r_{low}, r_{high} and tolerance t in reals, $t \geq 0.0$, if $\forall$ u_i in real array u, $r_{low} - t \leq u_i \leq r_{high} + t$.
sum(u)	$\sum u_i$, u_i in array u.

Handout Slide

As well as four O(n) relational-based assertion functions.

Given r in Relation Expressions
o in Relational Operators (i.e., <, <=, >, >=, ==, !=)
u, v in Arrays
n in Integers or Reals

$$r ::= u\ o\ n \mid n\ o\ u \mid u\ o\ v$$

Where $u\ o\ v$ is evaluated as $u_i\ o\ v_j\ \forall\ u_i$ in u, v_i in v

Function	Returns…
all(r)	True if r is satisfied $\forall$ u_i and, if appropriate, v_i.
any(r)	True if there exists u_i and, if appropriate, v_i such that r is satisfied.
count(r)	Total number of u_i such that r is satisfied for u_i and, if appropriate, v_i.
none(r)	True if there exists *no* u_i and, if appropriate, v_i such that r is satisfied.

Handout Slide

norm() can throw pre- & post-condition violation exceptions.

```
package vector version 1.0 {
  class Utils { …

    static double norm(in array<double> u, in double tol, in int badLevel)
      throws
        sidl.PreconditionViolation, NegativeValueException,
        sidl.PostconditionViolation;
      require
        not_null : u != null;
        u_is_1d : dimen(u) == 1;
        non_negative_tolerance : tol >= 0.0;
      ensure
        no_side_effects : is pure;
        non_negative_results : result >= 0.0;
        nearEqual(result, 0.0, tol) iff isZero(u, tol);
  … }
}
```

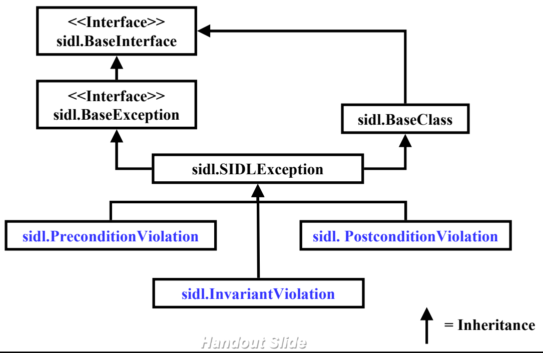
Implicit!

Preconditions

Postconditions

vector.sidl

Each type of assertion has own built-in exception.



<<Interface>> sidl.BaseInterface
<<Interface>> sidl.BaseException
sidl.BaseClass
sidl.SIDLException
sidl.PreconditionViolation
sidl. PostconditionViolation
sidl.InvariantViolation
= Inheritance

Handout Slide

The implementation also prints a message to an output file.

```
…
void impl_vector_Utils__check_error_static( /* in */ const char* msg)  {
  /* DO-NOT-DELETE splicer.begin(vector.Utils._check_error_static) */
  printMessage(msg);
  /* DO-NOT-DELETE splicer.end(vector.Utils._check_error_static) */
}
  …
void
impl_vector_Utils__check_error ( /* in */ vector_Utils self,
                                 /* in */ const char* msg)
{
  /* DO-NOT-DELETE splicer.begin(vector.Utils._check_error) */
  printMessage(msg);
  /* DO-NOT-DELETE splicer.end(vector.Utils._check_error) */
}
…
```

vector_Utils_Impl.c

Enforcement options (sidlAsserts.h)

```
<p>_<c>__set_checking_static(int32_t level, double rate, int32_t resetCounters);
<p>_<c>__set_checking (<c> self, int32_t level, double rate, int32_t resetCounters);
```

Boolean used to reset countdown.

Frequency (Periodic), Range (Random), Overhead limit (Timing).

Parts	Level options
Assertion Types	CHECK_NO_TYPES CHECK_PRECONDITIONS CHECK_POSTCONDITIONS CHECK_INVARIANTS CHECK_PRE_POST_ONLY CHECK_PRE_INV_ONLY CHECK_POST_INV_ONLY CHECK_ALL_TYPES
Assertion Checking (Policy)	CHECK_ALWAYS CHECK_PERIODICALLY CHECK_RANDOMLY CHECK_TIMING

Will be splitting level arg. into *types* and *policy*.

The checking level is currently set through bit op.

```
…
 /* Always check all three types of assertions (invariants, pre- and post-conditions */
 vector_Utils__set_checking_static(CHECK_ALL_TYPES | CHECK_ALWAYS, 0, 0);
…
 /* Always check postconditions only */
 vector_Utils__set_checking_static(CHECK_POSTCONDITIONS | CHECK_ALWAYS,
                                   0, 0);
…
 /* Always check preconditions only */
 vector_Utils__set_checking_static(CHECK_PRECONDITIONS | CHECK_ALWAYS,
                                   0, 0);
…
 /* Disable assertion checking */
 vector_Utils__set_checking_static(CHECK_NO_TYPES, 0, 0);
…
```

vectortest.c

PROTOTYPE

Babel Assertion and Method Hook Basics

- Assertion support
  - Babel options
  - SIDL grammar
  - Assertion functions
  - Assertion violation actions
  - Enforcement options
- Pre/Post method hooks
  - Babel options
  - Generated server
  - Client code

Hook generation can be enabled at compile-time.

- Disabled by default
- Long version

```
$ babel --server=c \
  --repository-path=$(REPO) \
  --generate-hooks hooks.sidl
```

- Short version

```
$ babel -sc -R$(REPO) -i hooks.sidl
```

hooks.sidl example from regression tests.

```
package hooks version 1.0
{
  class Basics {
    static void aStaticMeth(in double val);
    void aNonStaticMeth(in int val);
  }
}
```

hooks.sidl

hooks_Basics_Impl.c – static method

```
…
void impl_hooks_Basics_aStaticMeth_pre( /* in */ double val) {
  /* Prints "aStaticMeth_pre: " followed by val. */
}
…
void impl_hooks_Basics_aStaticMeth( /* in */ double val) {
  /* Prints "aStaticMeth: " followed by val. */
}
…
void impl_hooks_Basics_aStaticMeth_post( /* in */ double val) {
  /* Prints "aStaticMeth_post: " followed by val. */
}
```

hooks_Basics_Impl.c

hooks_Basics_Impl.c – non-static method

```
…
void impl_hooks_Basics_aNonStaticMeth_pre( /* in */ hooks_Basics self,
                                            /* in */ int32_t val) {
  /* Prints "aNonStaticMeth_pre: " followed by val. */
}
…
void impl_hooks_Basics_aNonStaticMeth( /* in */ hooks_Basics self,
                                       /* in */ int32_t val) {
  /* Prints "aNonStaticMeth: " followed by val.  */
}
…
void impl_hooks_Basics_aNonStaticMeth_post( /* in */ hooks_Basics self,
                                             /* in */ int32_t val) {
  /* Prints "aNonStaticMeth_post: " followed by the val. */
}
```

hooks_Basics_Impl.c

Dynamic hook options

```
<pkg>_<class>__set_hooks (int32_t on);
 <pkg>_<class>__set_hooks_static (<class> self, int32_t on);
```

Option	Options for on argument
Enable	1 (TRUE)
Disable	0 (FALSE)

Example driver from the regression test suite.

```
…
 hooks_Basics__set_hooks_static(TRUE); // Enable static pre/post


 hooks_Basics h = hooks_Basics__create();
 hooks_Basics__set_hooks(h, TRUE); // Enable non-static pre/post
 hooks_Basics_aStaticMeth(dVal);
 hooks_Basics_aNonStaticMeth(h, iVal);

 hooks_Basics_deleteRef(h);

 return 0;
}
```

hookstest.c

```
aStaticMeth_pre: <dVal>
aStaticMeth: <dVal>
aStaticMeth_post: <dVal>
aNonStaticMeth_pre: <iVal>
aNonStaticMeth: <iVal>
aNonStaticMeth_post: <iVal>
```

Output file

Babel Assertion and Method Hook Basics

PROTOTYPE

The End