Gem #149: Говорить правду, но (возможно) не всю

Автор: Yannick Moy — AdaCore

Краткое содержание: В Ada 2012 утверждение необходимых параметров программы не ограничены прагмой `Assert`. В данном Gem описывается использование прагмы `Assertion_Policy`, с помощью которой можно определять, какие утверждение должны выполняться в определенные отрезки времени выполнения

Давайте начнём…

В начале был создан язык программирования Ada. У него не было никаких утверждений. Затем пришел GNAT, который представил **`pragma`** `Assert`. ARG увидел, что это хорошо, и приняла его в Ada 2005. Потом пришел GNAT снова, в который введена **`pragma`** `Precondition` и **`pragma`** `Postcondition`. ARG увидел, что они тоже хороши, и принял их как аспекты в Ada 2012. ARG даже пытался победить GNAT в этой игре и одновременно ввел аспекты для предикатов типа (см. Gems #146 и #147) и инвариантов типа (см. Gem #148), которые являются другими формами утверждений. Потом снова GNAT представил прагмы: **`pragmas`** `Assume`, `Assert_And_Cut` и `Loop_Invariant`, и аспект `Contract_Cases`, а также и другие формы заявлений.

Таким образом, теперь у программиста Ada есть богатый набор утверждений для определения свойств, относящихся к управлению (`Assert`, `Pre`, `Post`, `Loop_Invariant`, `Assume`, `Assert_And_Cut`) и свойств, относящихся к данным (`Static_Predicate`, `Dynamic_Predicate`, `Type_Invariant`).

Как определить, какие утверждения выполняются? И как различать различные исполняемые файлы, скажем, между одним, созданным для отладки/тестирования, и одним, созданным для производства?

GNAT предоставляет ключ `-gnata`, который активирует все утверждения: конечно включает **`pragma`** `Assert`, но также и все новые формы утверждений, представленные выше. Таким образом, каждый блок может быть независимо скомпилирован с утверждениями или без них. Но это не всегда достаточно и правильно.

Давайте возьмем пример написания библиотеки. Мы хотим использовать предварительные условия, чтобы предотвратить вызов библиотеки в недопустимом контексте (защитное программирование), и постусловия плюс предикаты типа, чтобы помочь с отладкой и обслуживанием библиотеки (проверка на основе утверждений). Вот код:

```
package Library is
   type Status is (None, Acquired, Released);

   type Resource is record
      Id   : Integer;
      Stat : Status;
   end record
     with Dynamic_Predicate =>
       (if Resource.Id = 0 then Resource.Stat = None
        else Resource.Stat /= None);

   No_Resource : constant Resource := Resource'(0, None);

   procedure Get (R : in out Resource; Id : Integer) with
     Pre  => R.Stat = None,
```

```
      Post => R.Stat = Acquired;

   procedure Free (R : in out Resource) with
      Post => (if R.Stat'Old = Acquired then R.Stat = Released);
end Library;


package body Library is
   procedure Get (R : in out Resource; Id : Integer) is
   begin
      R.Stat := Acquired;
      R.Id   := Id;
   end Get;

   procedure Free (R : in out Resource) is
   begin
      if R.Stat /= Acquired then
         return;
      end if;
      R.Stat := Released;
   end Free;
end Library;
```

Когда этот код компилируется с ключом `-gnata`, каждый вызов `Get` вызывает четыре утверждения во время выполнения (и три при вызове `Free`):

- проверка предварительных условий при входе в подпрограмму
- проверка постусловия при выходе из подпрограммы
- проверка предиката для параметра `R` при вводе подпрограммы
- проверка предиката для параметра `R` при выходе из подпрограммы

Это хорошо во время тестирования и отладки (когда мы используем `-gnata`), но мы бы хотели, чтобы производственный код содержал только утверждения времени выполнения для предварительных условий, чтобы отследить неправильное использование библиотеки в реальном продукте, избегая при этом накладных расходов другие проверки.

Ada 2012 предоставляет прагму **`pragma`** `Assertion_Policy` для этой цели. Эта прагма может принимать имя аспекта утверждения/прагмы в качестве первого аргумента и желаемую политику для этого аспекта в качестве второго аргумента. Чтобы обеспечить проверку предварительных условий, даже если `-gnata` не используется, нужно только включить следующую строку в начале файла `library.ads`:

```
pragma Assertion_Policy (Pre => Check);
```

Теперь будет обнаружено любое неправильное использование библиотеки клиентским кодом, независимо от того, как библиотека скомпилирована. Возьмем, к примеру, программу, которая не может освободить ресурс между двумя вызовами `Get`:

```
with Library; use Library;
procedure Client is
   R : Resource := No_Resource;
begin
   Get (R, 1);
```

```
   Get (R, 2);  -- incorrect
end Client;
```

Этот код (и код библиотеки) теперь можно скомпилировать без `-gnata`:

```
$ gnatmake client.adb
gcc -c client.adb
gcc -c library.adb
gnatbind -x client.ali
gnatlink client.ali
```

И этот код все еще вызывает ошибку во время выполнения:

```
$ ./client
raised SYSTEM.ASSERTIONS.ASSERT_FAILURE : failed precondition from library.ads:16
```

Для получения дополнительной информации о прагме `Assertion_Policy` или о новых утверждениях – прагмах/аспектах, поддерживаемых GNAT, см. Справочное руководство по GNAT Pro.

И как говорит Тони Хоар (Tony Hoare): «Утверждай рано и утверждай часто!»

Связанный со статьёй текст программы

Файлы примеров Ada Gems распространяются AdaCore и могут быть использованы или изменены для любых целей без ограничений.

Об авторе

Yannick Moy — AdaCore

Работа Yannick Moy сосредоточена на анализе исходного кода программного обеспечения, в основном для обнаружения ошибок или проверки свойств безопасности / обеспечения безопасности. Yannick ранее работал в PolySpace (теперь The MathWorks), где он начал проект C ++ Verifier. Затем он поступил в INRIA Research Labs / Orange Labs во Францию для проведения PhD по автоматической модульной проверке статической безопасности для программ на C. Янник присоединился к AdaCore в 2009 году, после короткой стажировки в Microsoft Research.

Yannick Moy имеет степень инженера из Политехнической школы (Ecole Polytechnique), магистра в Стэнфордском университете (Stanford University) и доктора философии из Университета Парижа-Суд (Université Paris-Sud). Он является учёным Siebel.

Last Updated: 10/13/2017
Posted on: 6/3/2013

Обсуждение…