Gem #148: Подтипы в Ада 2012 - Часть 3

Автор: Yannick Moy — AdaCore

Краткое содержание: В предыдущих Gem серии мы показали, как использовать `Static_Predicate` и `Dynamic_Predicate` для определения неизменяемых параметров объектов. Тема данного Gem - аспект `Type_Invariant`.

Давайте начнём…

В предыдущих двух Gems мы видели, как аспекты `Static_Predicate` и `Dynamic_Predicate` могут использоваться для определения свойств объектов, которые должны соблюдаться всегда. Этот третий и последний камень в серии посвящен аспекту, который называется `Type_Invariant`.

Аспект `Type_Invariant` может использоваться с закрытыми типами, чтобы определить свойство, которое все объекты типов должны учитывать за пределами пакета, где типы объявлены. Возьмем, например, тип `Communication`, хранящий сообщения между различными сторонами, на основе типа сообщения, использованного в предыдущем Gem:

```
package Communications is
   type Message_Arr is array (Integer range <>) of Message;
   type Communication (Num : Positive) is private;
private
   type Communication (Num : Positive) is record
      Msgs : Message_Arr (1 .. Num);
   end record;
end Communications;
```

Чтобы указать, что сообщения должны быть упорядочены по дате получения, мы можем добавить аспект к полному типу:

```
type Communication (Num : Positive) is record
   Msgs : Message_Arr (1 .. Num);
end record with
  Type_Invariant =>
    (for all Idx in 1 .. Communication.Num-1 =>
      Communication.Msgs(Idx).Received <= Communication.Msgs(Idx+1).Received);
```

Компилятор вставит проверки выполнения, чтобы убедиться, что это свойство хранится в заданных местах кода:

- при инициализации объекта (в том числе по умолчанию!)
- о преобразованиях в тип
- при возврате объекта из публичной функции, определенной в пакете типа
- по параметрам out и in out при возврате из публичной процедуры пакета типа

Например, рассмотрим следующий неправильный код, который не может инициализировать `Com` до правильного значения, удовлетворяющего инварианту:

```
Com : Communication (2);  -- incorrect
```

Компиляция его с утверждениями и запуск приводит к следующей ошибке:

```
raised SYSTEM.ASSERTIONS.ASSERT_FAILURE : failed invariant from communications.ads:16
```

Но если мы дадим объекту явное значение через функцию создания `Create`, определенную в модуле `Communications`, то объявление объекта будет разработано без ошибок:

```
Coms : Communication (2) := Create (A);
```

Внутри функции `Create` инициализация `Coms` должна учитывать инвариант, но после этого инвариант может быть нарушен между временем объявления `Coms` и временем его возврата.

```
function Create (A : Message_Arr) return Communication is
   Coms : Communication := (Num => A'Length, Msgs => A);
begin
   -- statements before the return might violate the invariant
   return Coms;
end Create;
```

Ada требует, чтобы инвариант типа проверялся на каждой части параметра, имеющего связь типа, где часть может быть компонентом записи, или элементом массива, или любой такой комбинацией. Например, он проверяется на каждом элементе массива, возвращаемом `Create_N` или, потенциально измененном `Update_N`:

```
type Communication_Arr is array (Integer range <>) of Communication;
function Create_N return Communication;
procedure Update_N (A : in out Communication_Arr);
```

Важно отметить, что инвариант не проверяется в подпрограммах, объявленных в приватной части или в теле пакета. Эти подпрограммы являются внутренними операциями и должны вызываться для объектов, инвариант которых не выполняется. Аналогично, инвариант не проверяется для параметров режима `in`, например, для функций запроса, используемых в определении самого инварианта типа. Это правильный подход, так как в противном случае это легко привело бы к бесконечным циклам!

В качестве примечания стоит упомянуть, что GNAT также предоставляет аспект с именем `Invariant`, который является синонимом аспекта `Type_Invariant` (и реализован до появления `Type_Invariant` в Ada 2012).

Этот Gem заканчивает серию из трех Gem на тему «Подтипы в Ада 2012». Вместе со `Static_Predicate` и `Dynamic_Predicate`, `Type_Invariant` предоставляет новые способы для определения свойств ваших данных, как в новых, так и в существующих программах, так что попробуйте их!

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: 5/20/2013

Обсуждение…