Gem #146: Подтипы в Ада 2012 - Часть 1

Автор: Yannick Moy — AdaCore

Краткое содержание: Новая версия языка Ada полна инструментов для определения параметров типов. В данной серии из трех публикаций мы опишем три аспекта, которые могут быть использованы для определения неизменяемых параметров типов. Тема первого Gem - аспект Static_Predicate.

Давайте начнём…

Ada 2012 полна возможностей для задания богатого набора свойств типа. В этой серии из трех статей Gem мы описываем три аспекта, которые могут быть использованы для определения инвариантных свойств типов и подтипов. Это первый Gem 146 этой серии статей касается `Static_Predicate` аспект.

`Static_Predicate` можно указать в скалярных типах и определениях подтипов, чтобы указать свойство, которое все объекты подтипа должны соблюдать все время. Возьмем, например, тип `Day`, представляющий дни недели:

```
type Day is (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday);
```

Чтобы указать, что `T_Day` - это (sub)тип дней, имя которых начинается с `'T'` , мы можем написать:

```
type T_Day is new Day with Static_Predicate => T_Day in Tuesday | Thursday;
```

или

```
subtype T_Day is Day with Static_Predicate => T_Day in Tuesday | Thursday;
```

Теперь компилятор предупредит о программе, которая назначает статически известное значение, отличающееся от вторника или четверга, для объекта `T_Day`.

Мы продолжим с использованием второго определения, приведенного выше по тексту. Например, на этом неправильном коде:

```
D : T_Day := Day'First; -- Incorrect
```

GNAT генерирует следующее предупреждение во время компиляции:

```
>>> warning: static expression fails static predicate check on "T_Day"
```

Компилятор также проверяет полноту выражений `case` и операторов `case` с аргументами `T_Day`. Например, по этому коду:

```
   case D is
      when Tuesday => ...
      when Friday => ...  -- Incorrect
   end case;
```

GNAT генерирует следующие ошибки во время компиляции:

```
>>> missing case value: "Thursday"
>>> static predicate on "T_Day" excludes value "Friday"
```

Если пятница заменяется правильным значением четверг, то код компилируется спокойно.

Наконец, компилятор генерирует проверки во время выполнения для любой ошибочной записи дня, отличного от вторника или четверга в объекте типа `T_Day`, что позволяет легко обнаружить нарушения предиката типа, как только это происходит! Обратите внимание, что для включения проверки `Static_Predicate` во время выполнения (и других видов утверждений, указанных аспектами) необходимо скомпилировать с помощью `switch-gnata` (или включить проверку утверждений с помощью `Pragma Assertion_Policy`).

Например, предположим, что у нас есть процедура `Next`, которая продвигает свой аргумент на следующий день, и мы хотим определить аналогичную процедуру `Next_T`, которая продвигает свой аргумент подтипа `T_Day`. Вот определение дня процедуры:

```
procedure Next (D : in out Day) is
begin
   if D = Sunday then
      D := Monday;
   else
      D := Day'Succ (D);
   end if;
end Next;
```

Ниже приведена неудачная попытка определения Next_T:

```
procedure Next_T (D : in out T_Day) is
begin
   Next (D); -- Incorrect
   while D not in T_Day loop
      Next (D);
   end loop;
end Next_T;
```

Давайте добавим тест этого кода:

```
with Days; use Days;
procedure Main is
   D : T_Day := Tuesday;
begin
   Next_T (D);
end Main;
```

Когда этот код компилируется с включенными утверждениями `(- gnata)` и запускается, он выдает ошибку времени выполнения:

```
raised SYSTEM.ASSERTIONS.ASSERT_FAILURE : Static_Predicate failed at days.adb:3
```

Это указывает на первую строку, где `Next` вызывается в `Next_T`. Действительно, при входе в `Next_T` значение `D` равно `Tuesday`, поэтому `Next` возвращает среду, которая не удовлетворяет `Static_Predicate T_Day`, но назначается `T_Day`, что вызывает ошибку времени выполнения. Правильная версия `Next_T` использует временную переменную типа `T_day Basis`, которая удаляет все ограничения из `T_Day`, включая предикат, если он присутствует:

```
procedure Next_T (D : in out T_Day) is
   Tmp : T_Day'Base := D;
begin
   Next (Tmp);
```

```
   while Tmp not in T_Day loop
      Next (Tmp);
   end loop;
   D := Tmp;
end Next_T;
```

В следующем Gem этой серии мы увидим, как использовать связанный аспект с именем `Dynamic_Predicate`.

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: 4/22/2013

Обсуждение…

6 Comments

- **Peter Hermann**

Apr 23rd, 2013

well explained, kudos.

Moreover I am happy with the Monday as the first day of the week.

- **Yannick Moy**

Apr 25th, 2013

Thanks Peter. I had to ask some colleagues in the US what was so special about Monday being the first day of the week, and it turns out this is not a universally shared fact!

But note that The One True standard about dates, ISO 8601, defines Monday as the first day of the week. The following pages to provide interesting discussion on this matter:

http://www.timeanddate.com/calendar/days/
http://en.wikipedia.org/wiki/Sunday

- **Daniel Bigelow**

May 3rd, 2013

To explain the new concepts and mechanisms introduced into Ada-2012, which enhance the language even further in support of the primary objective - the early detect software defects - it was apparently necessary to increase the vocabulary of the technology to an even higher level of sophistication then it already is. So now in addition to rendezvous, aggregate, progenitor, variant, attribute, pragma, aliasing, volatile, discriminant , scalar, affinity, etc . . . we now have aspects, invariants, and predicates. What is this, Shakespeare?!

It seems that even the academics in the field, (i.e. computer scientists, not Shakespearians) are starting to trip over some of these words. For example, you made the statement, “In this series of three Gems, we describe three aspects that can be used to state invariant properties of types and subtypes. “. Say what? That doesn’t make sense; only one aspect of the three can be used to state that a type is “Invariant”: the so-called “Type_Invariant”. The properties discussed in this GEM concern static and dynamic “Predicates” that generalize the concept of a range constraint. I know what you wanted to say. I am just pointing out that the wording is confusing. I would have either left out the word “invariant” or replaced it with “unchangeable” to avoid confusion with the aspect “Type_Invariant”.

Also, you stated that, “Ada 2012 is full of features for specifying a rich set of type properties.” The feature you are referring to of course is “aspect specifications”. The operative word here is “Properties”. Until recently it was not really necessary to apply this word in the Ada context since we had another word for the same idea, namely, “Attribute” which is understood to be a “getter” operation that returns information about the property of a type or an object. However, now that we can add properties to types via the aspect mechanism I think the time has come to make a clear distinction between the words attribute and property.

Bottom line: Before the publication of “Programming in Ada 2012, by John Barnes” goes to press it needs to include an expanded Glossary in which the new terms (including Property) are described in detail.

Looking forward to part 2 in the series.

- **Yannick Moy**

May 3rd, 2013

Hi Daniel,

You’re right that the mention of “invariant properties” could be confusing here. I meant invariant in its usual meaning, not as a reference to Type_Invariant (which will be discussed in part 3).

Regarding the specification features, I meant of course the aspects presented in this series of Gems, but also the richer set of expressions in Ada 2012 that make it easier to express complex properties in these aspects. And I used the word “properties” in its usual meaning, to state that really these new aspects allow you to refine the set of abstract properties of a datatype.

I’m sure the book by John Barnes will contain a glossary, but I don’t know if it defines the abstract notion of “property”. But you got the idea here!

- **David Mentré**

May 22nd, 2013

Isn’t there an error in the sentence “Indeed, on entry to Next_T, the value of D is Tuesday, so Next returns Wednesday, which does satisfy the Static_Predicate of T_Day, but is assigned to a T_Day, hence triggering a run-time error.”?

I think it should be: “Indeed, on entry to Next_T, the value of D is Tuesday, so Next returns Wednesday, which does *not* satisfy the Static_Predicate of T_Day *and* is assigned to a T_Day, hence triggering a run-time error.”

- **Yannick Moy**

May 22nd, 2013

@David:

You’re right, there was a missing negation, which kind of changes the meaning… I have corrected the text of the Gem. (I’ve left *but* instead of *and*, the idea being the same, that this assignment is not allowed.) Thanks!