Gem #136: Сколько метров в килограмме?

Автор: Vincent Pucci—AdaCore

Краткое содержание: В компилятор GNAT была добавлена возможность проверки размерности. Пользователь может определить физическую величину для объекта, и компилятор будет проверять совместимость объектов с их размерностью способом, схожим с таковым в инженерной практике. Измерения алгебраических выражений (включая мощности со статическими экспонентами) рассчитывается по их составляющим. В компиляторе GNAT доступен пакет поддержки системы МКС (метр-килограмм-секунда; *прим.* как часть т. н. система “СИ”), и при необходимости пользователь может добавлять дополнительные пакеты (сантиметр-грамм-секунда, эмпирические единицы, и т.п.)

Давайте начнём…

Этот Gem описывает новую систему проверки размерности GNAT. Эта функция опирается на спецификации аспектов Ada 2012 и доступна начиная с версии 7.0.1 GNAT.

Компилятор GNAT теперь поддерживает проверку размерности. С тех пор, как в Ada 83 появились пользовательские операторы, были попытки использовать их для объявления типов с измерениями и выполнения проверок размерности в научном коде. Эти попытки были, к сожалению, громоздкими и не были приняты языком. Система, которую мы здесь описываем, легка и основана на спецификациях аспектов, представленных в Ada 2012. Мы надеемся, что инженерное и научное сообщество найдет ее удобным и простым в использовании.

Прежде чем исследовать реализацию этой функции, давайте сначала рассмотрим общий пример в физике: проблема свободного падения. Мы хотим оценить расстояние, пройденное телом за 10 секунд свободного падения, зная ускорение гравитации на Земле. Хорошо известным решением является:

```
distance = 1/2 * g * t**2 where t = 10s.
```

Здесь возникает вопрос: как мы определяем размерность объектов в Ada?

Для этой цели компилятор GNAT предоставляет Международную систему единиц, то есть систему единиц метр-килограмм-секунда (MKS), в пакете `System.Dim.Mks`. GNAT также предоставляет средства вывода для этой системы модулей в пакете `System.Dim.Mks_IO`.

Пакет `System.Dim.Mks` определяет числовой базовый тип `Mks_Type`, и определяются его подтипы, которые соответствуют физическим величинам (таким как длина, время), а также константы каждой измеренной величины (такой как `m`, `s`).

Используя эти два пакета, мы теперь можем преобразовать нашу проблему свободного падения в код Ada:

```
with System.Dim.Mks; use System.Dim.Mks;
with System.Dim.Mks_IO; use System.Dim.Mks_IO;
with Text_IO; use Text_IO;

procedure Free_Fall is

   G        : constant Mks_Type := 9.81 * m / (s**2);
   T        : Time := 10.0*s;
   Distance : Length;
```

```
begin
   Put ("Gravitational constant: ");
   Put (G, Aft => 2, Exp => 0); Put_Line ("");
   Distance := 0.5 * G * T ** 2;
   Put ("distance traveled in 10 seconds of free fall: ");
   Put (Distance, Aft => 2, Exp => 0);
   Put_Line ("");
end Free_Fall;
```

Компиляция и запуск нашего примера дает:

```
$ gnatmake -q -gnat2012 free_fall.adb
$ ./free_Fall
Gravitational constant: 9.81 m.s**(-2)
distance traveled in 10 seconds of free fall: 490.50 m
```

Свойства типа `Mks_Type` и его подтипов `Time` и `Length` определяются с помощью двух новых специфичных для GNAT аспектов:

Аспект `Dimension_System` позволяет пользователю определить систему единиц. Аспект `Dimension_System` может применяться только к числовому типу, обычно к типу с плавающей запятой соответствующей точности (любой числовой тип может использоваться в качестве базового). Тип (точнее, первый подтип), к которому применяется аспект `Dimension_System`, является размерным типом. Синтаксис аспекта `Dimension_System` выглядит следующим образом:

```
with Dimension_System => ( DIMENSION {, DIMENSION});

DIMENSION ::= ( [Unit_Name =>] IDENTIFIER, [Unit_Symbol =>] SYMBOL, [Dim_Symbol =>] SYMBOL)

SYMBOL ::= STRING_LITERAL | CHARACTER_LITERAL
```

То есть значение аспекта является агрегатом. Могут быть определены до 7 размеров (соответствующий Международной системе единиц).

Аспект размерности позволяет пользователю объявлять определенные размерности величин количества в данной системе. Аспект размерности применяется только к подтипу определенного `Dimension` типа. Подтип, к которому применяется аспект `Dimension`, является определенным подтипом размерности. Синтаксис аспекта `Dimension` следующий:

```
with Dimension => ( [[Symbol =>] SYMBOL,] DIMENSION_VALUE {, DIMENSION_VALUE});

SYMBOL ::= STRING_LITERAL | CHARACTER_LITERAL

DIMENSION_VALUE ::= RATIONAL | others => RATIONAL | DISCRETE_CHOICE_LIST => RATIONAL

RATIONAL ::= [-] NUMERAL [/NUMERAL]
```

Используя оба аспекта, пользователи таким образом в состоянии определить свои собственные системы единиц и вычислить определенные размерности формулы. Компилятор диагностирует несоответствие размерности в вычислениях и присвоениях.

Теперь, давайте более внимательно рассмотрим пакет MKS. MKS определяет тип определенной размерности следующим образом:

```
type Mks_Type is new Long_Long_Float
```

```
with
 Dimension_System => (
   (Unit_Name => Meter,    Unit_Symbol => 'm',   Dim_Symbol => 'L'),
   (Unit_Name => Kilogram, Unit_Symbol => "kg",  Dim_Symbol => 'M'),
   (Unit_Name => Second,   Unit_Symbol => 's',   Dim_Symbol => 'T'),
   (Unit_Name => Ampere,   Unit_Symbol => 'A',   Dim_Symbol => 'I'),
   (Unit_Name => Kelvin,   Unit_Symbol => 'K',   Dim_Symbol => "Theta"),
   (Unit_Name => Mole,     Unit_Symbol => "mol", Dim_Symbol => 'N'),
   (Unit_Name => Candela,  Unit_Symbol => "cd",  Dim_Symbol => 'J'));
```

Подтипы определенной размерности (такие как `Length`, `Mass`, `Time`, `Electric_Current`, `Thermodynamic_Temperature`, `Amount_Of_Substance`, `Luminous_Intensity`) определяются следующим образом:

```
subtype Length is Mks_Type
  with Dimension => (Symbol => 'm',
                     Meter  => 1,
                     others => 0);
```

Пакет также определяет условные имена для значений каждой единицы, такие как:

```
m  : constant Length := 1.0;
cm : constant Length := 1.0E-02;
```

В аспектах `Dimension` и `Dimension_System` символы, передаваемые в качестве аргументов, используются для целей вывода. Действительно, GNAT также предоставляет специфичные для измерения средства вывода в двух универсальных пакетах для целочисленных и плавающих типов: `System.Dim.Integer_IO.` и `System.Dim.Float_IO.` Процедуры `Put`, определенные в этих пакетах, выводят числовое значение, за которым следует вектор измерения элемента, переданный в качестве параметра (`System.Dim.Mks_IO`, используемый в примере свободного падения, является экземпляром `System.Dim.Float_IO`, экземпляром которого является `Mks_Type` в пакете `System.Dim.Mks`).

Возвращаясь к примеру со свободным падением, внимательный читатель заметит, что гравитационная постоянная `G` может быть определена путем введения нового подтипа размерности:

```
subtype Acceleration is Mks_Type
  with Dimension => ("m/sec^2
                     Meter  => 1,
                     Second => -2,
                     others => 0);
G : constant Acceleration := 9.81 * m / (s**2);
```

Выполнение программы `free_fall` теперь дает:

```
$ ./free_fall
Gravitational constant: 9.81 m/sec^2
distance traveled in 10 seconds of free fall: 490.50 m
```

Целью анализа измерений является проверка согласованности измерений в физических отношениях и предотвращение появления легендарных ошибок, таких как космический зонд, который был потерян из-за того, что один модуль был вычислен в метрических единицах, а другой - в эмпирических единицах. GNAT проверяет согласованность измерений на назначения, передачу параметров, операции сложения и сравнения. GNAT вычисляет результирующие измерения операций умножения и возведения в степень.

Например, в примере свободного падения неправильное назначение, такое как:

```
   Distance := 5.0 * kg;
```

отторгается во время компиляции со следующими диагнозами:

```
$ ./free_fall

free_fall.adb:15:13: dimensions mismatch in assignment

free_fall.adb:15:13: left-hand side has dimension [L]

free_fall.adb:15:13: right-hand side has dimension [M]
```

Неправильные арифметические операции также приводят к таким диагнозам. Например, в программе `free fall`:

```
   Distance := T + G;
```

отклоняется следующим образом:

```
$ ./free_fall

free_fall.adb:15:18: both operands for operation "+" must have same dimensions

free_fall.adb:15:18: left operand has dimension [T]

free_fall.adb:15:18: right operand has dimension [L.T**(-2)]
```

Этого введения должно быть достаточно, чтобы пользователь начал экспериментировать с существующими пакетами по инженерному/научному коду. Следующий шаг для амбициозного читателя – создать новую систему единиц, такую как система сантиметр-грамм-секунда (CGS), и протестировать как можно больше физических формул, чтобы ознакомиться с этой новой функцией. Если присутствует несколько систем, естественно будет ввести между ними процедуры преобразования. Поскольку базовые типы различных систем различны, проверка типов гарантирует, что эти значения не могут быть случайно смешаны в одном вычислении.

Это новая функция набора инструментальных средств GNAT. Мы приветствуем комментарии пользователей, предложения по улучшению, и интересные примеры использования!

Связанный со статьёй текст программы

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

Об авторе

Автор: Vincent Pucci - AdaCore

Last Updated: 10/13/2017
Posted on: 11/12/2012

Обсуждение…