AdaCore
The GNAT Pro Company

Abstract Interface Types in GNAT: Conversions, Discriminants and C++

Ada Europe'2006
Porto

Javier Miranda and Edmond Schonberg

Previous Papers

J.Miranda, E.Schonberg,G.Dismukes
"The Implementation of Ada 2005 Interface Types in the GNAT Compiler"
AdaEurope'2005

J.Miranda, E.Schonberg, H.Kirtchev
"The Implementation of Ada 2005 Synchronized Interfaces in the GNAT Compiler"
SigAda'2005

Contents

- **Multi-language programming: C++ and Ada**
- **Compiler support for abstract interface types**
  - Interface type conversions
  - GNAT Specific ABI support for discriminants

AdaCore
The GNAT Pro Company

Part I
Multi-Language Programming: Interfacing C++ and Ada 2005 with GNAT

Interface to other languages
Standard Ada support (ARM Annex B)

- **Pragmas**
  - Pragma Import
  - Pragma Export
  - Pragma Convention
  - Pragma Linker_Options
- **Packages**
  - Interfaces
  - Interfaces.C
  - Interfaces.C.Strings
  - Interfaces.C.Pointers
  - Interfaces.Cobol
  - Interfaces.Fortran

```
-- Map and use the following C function
--      void dump_data (int *ptr);

with Interfaces.C; use Interfaces;
procedure Binding_Example is
  procedure Dump_Data (Ptr : access C.Int);

  pragma Import
     (Convention => C,
      Entity          => Dump_Data,
      Link_Name  => "dump_data");

  Myint : aliased C.Int := 12;
begin
  Dump_Data (Myint'Access);
end;
```

Interfacing with C++

- **No support defined in the Ada 95 RM**
  - C++ had no standard until 1998
- **GNAT Specific support:**
  - Pragma CPP_Class
  - Pragma CPP_VTable
  - Pragma CPP_Virtual
  - Pragma CPP_Constructor

```
class My_Class {
    int Value;

    virtual void Set_Value  (int V)
    virtual int   Get_Value ();
    My_Class (); // Constructor
}
```

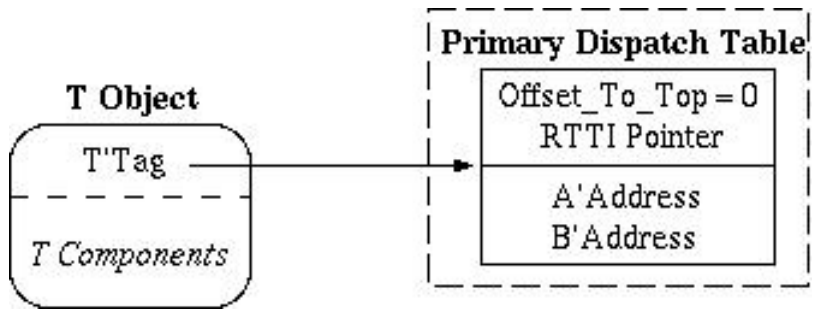
```
type My_Class is tagged record
    Vptr   : Interfaces.CPP.Vtable_Ptr;
    Value : Integer;
end record;
pragma CPP_Class   (Entity         => My_Class);
pragma CPP_Vtable (Entity         => My_Class,
                   Vtable_Ptr     => Vptr,
                   Entry_Count => 2);

procedure Set_Value (This : My_Class; V : Integer);
pragma CPP_Virtual (Set_Value, Vptr, 1);
pragma Import (CPP, Set_Value, "_ZN1A10SetValueEv");

function Get_Value (This : My_Class) return Integer;
pragma CPP_Virtual (Get_Value, Vptr, 2);
pragma Import (CPP, Get_Value, "_ZN1A10GetValueEv");

function New_Object return My_Class'Class;
pragma CPP_Constructor (New_Object);
pragma Import (CPP, Constructor, "_ZN1ANew_Object2Ev")
```

Interfacing with C++

- Ada 95 had single inheritance
- The Ada 2005 RM does not specify further support for interfacing with C++
- Ada 2005 ***ABSTRACT INTERFACE TYPES*** provides multiple inheritance of specifications

C++ ABI
GNU C++ Compiler

```
class T {
    // T components

    virtual void A ()
    virtual void B ();
}
```



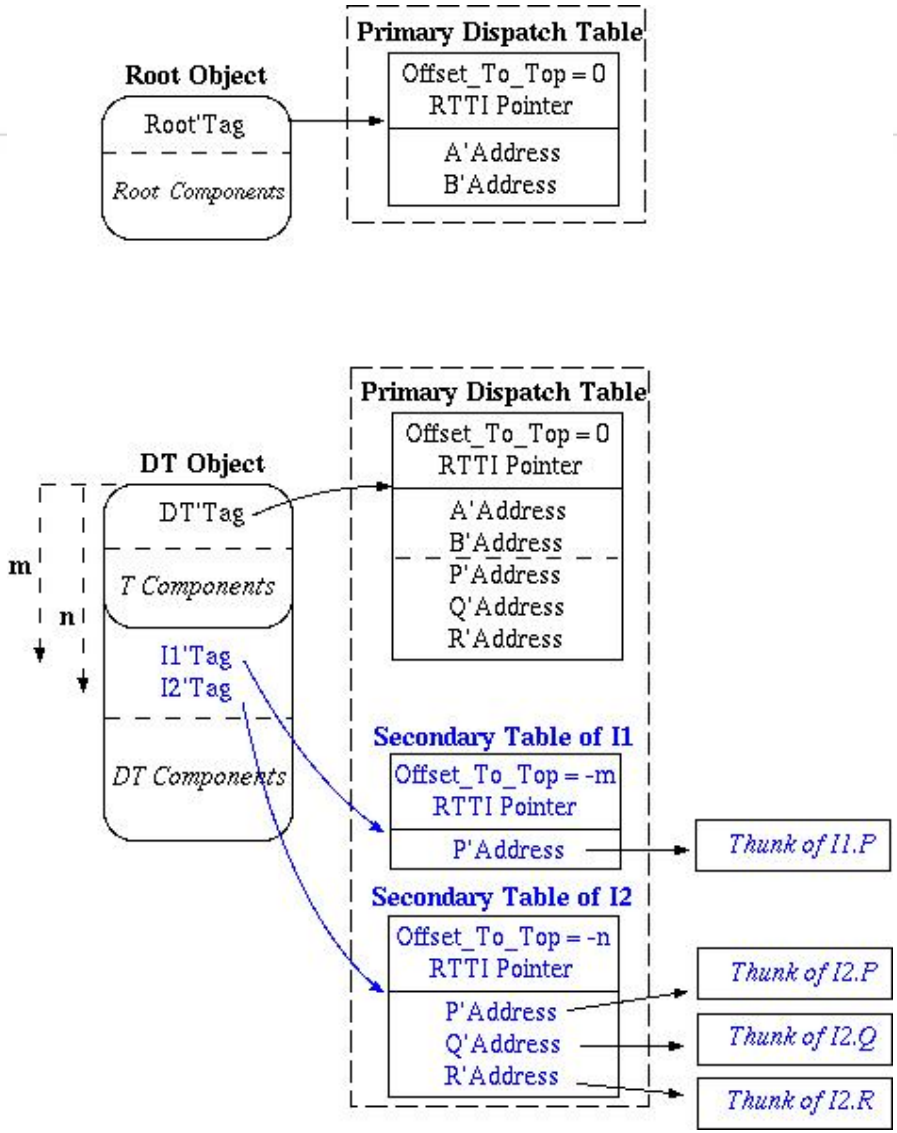
Itanium C++ Application Binary Interface
Code Sourcery, Compaq, EDG, HP, IBM, Intel, Red-Hat and SGI

C++ ABI
GNU C++ Compiler

```
class I1 {
    virtual void P () = 0;
}

class I2 {
    virtual void Q () = 0;
    virtual void R () = 0;
}
```

```
class Root {
    // Root components

    virtual void A ()
    virtual void B ();
}
```

```
class DT : Root, I1, I2 {
    // DT components

    virtual void P ()
    virtual void Q ();
    virtual void R ();
}
```



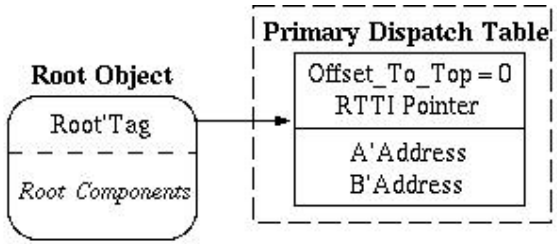
Itanium C++ Application Binary Interface
Code Sourcery, Compaq, EDG, HP, IBM, Intel, Red-Hat and SGI

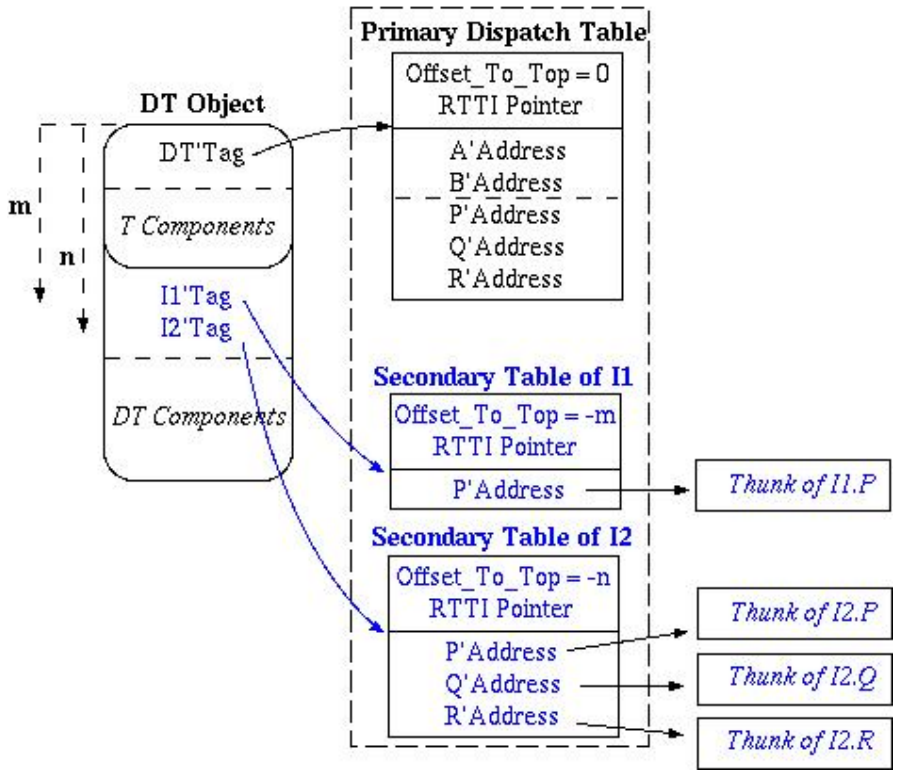
Compatibility at the ABI Level

```
type I1 is interface;
procedure P (Obj : I1) is abstract;

type I2 is interface;
procedure Q (Obj : I2) is abstract;
procedure R (Obj : I2) is abstract;
```

```
type Root is tagged record with
   -- Root components
   . . .
end record;

procedure A (Obj : Root);
procedure B (Obj : Root);
```

```
type DT is new Root and I1 and I2 with
   -- DT components
   . . .
end record;

procedure P (Obj : DT);
procedure Q (Obj : DT);
procedure R (Obj : DT);
```





Exporting to C++

- ***We can also extend an imported C++ class in the Ada side***

```
type Extension is new My_Class with
     More_Data : Integer;
end record;
pragma Convention (CPP, Extension);

procedure  Some_Further_Op (This : Extension);
pragma Export (CPP, Some_Further_Op);
```

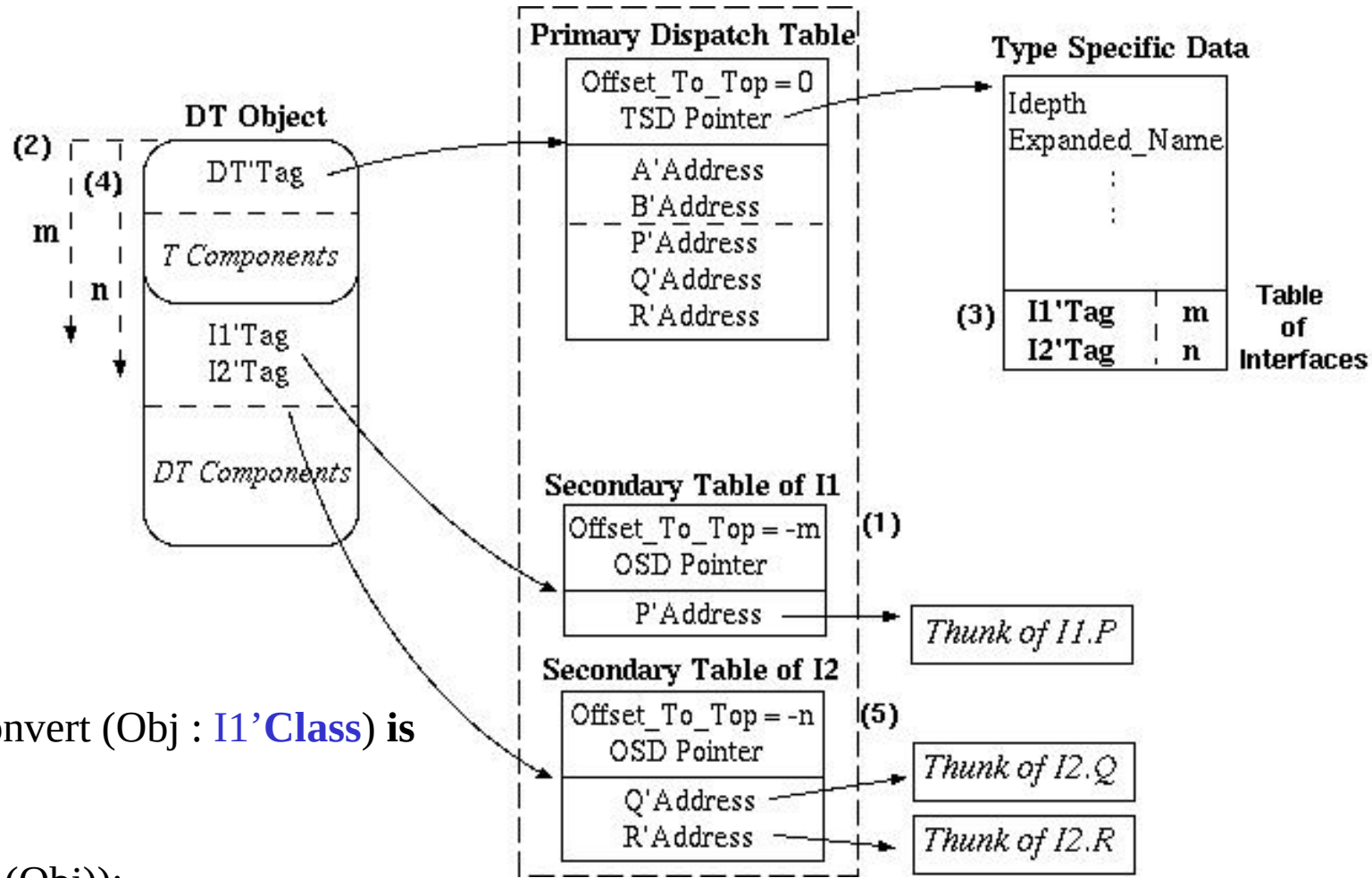
- ***The only requirement is that components must be declared in the same order***

Support available in GNAT Pro and the GAP 2006 release

AdaCore
The GNAT Pro Company

Part II
Abstract Type Conversions and Discriminants

Abstract Interface Type Conversion




```
procedure Convert (Obj : I1'Class) is
begin
   P (Obj);
   R (I2'Class (Obj));
end Convert;
```

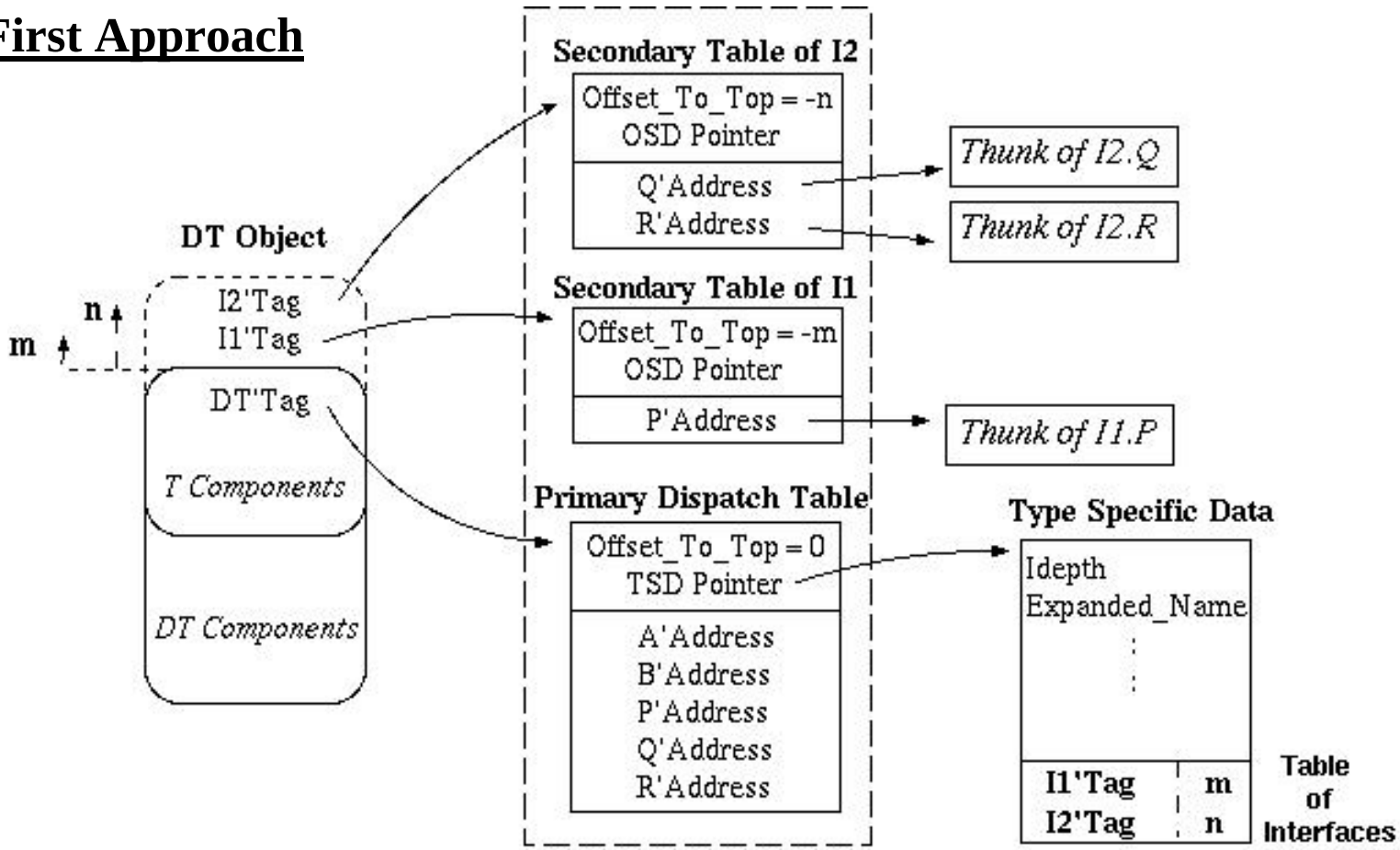
GNAT ABI Extension for Discriminants

```
type Root (Disc : Positive) is tagged record
    Name : String (1 .. Disc);
end record;

type DT is new Root and I1 and I2 with
    -- DT components
    . . .
end record;
```

The C++ ABI does not consider this case because C++ does not have non-static components.

GNAT ABI Extension for Discriminants

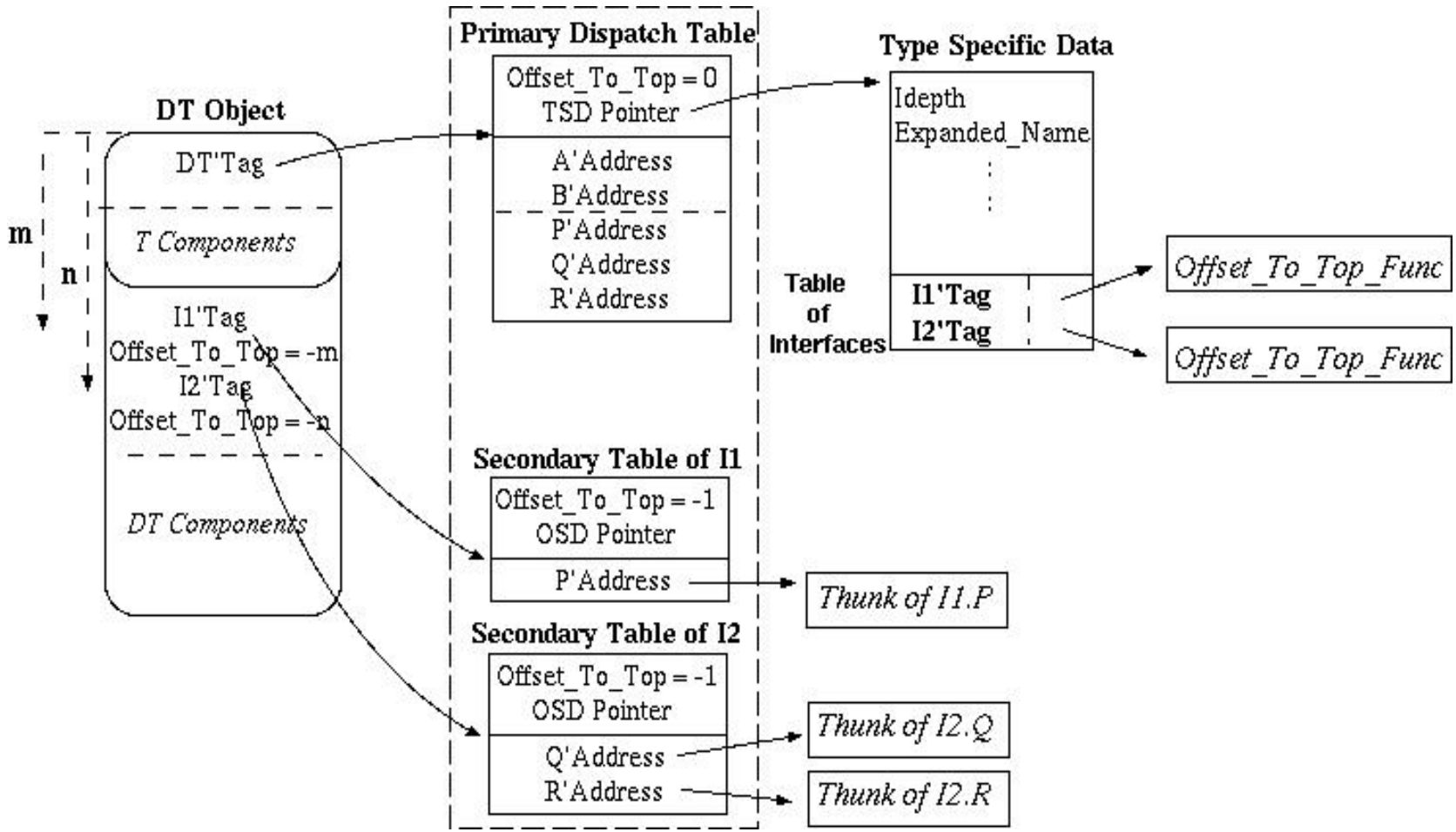
First Approach



Rejected because X'Address must denote the address of the first of its storage elements (ARM)

GNAT ABI Extension for Discriminants

Second Approach



Final Remarks

- **Interfacing with C++ at the class level**
  - Support already available in **GNAT Pro** and the **GAP 2006** release
  - An extended paper will be published soon in the GCC summit http://www.gccsummit.org/2006
  - Further documentation will be available soon in the AdaCore web server http://www.adacore.com
  - We are currently working in further support to automatically generate the C++ mangled names
- **Interface Conversions and Discriminants with Interfaces**
  - Support already available under **GNAT Pro** and the **GAP 2006** release