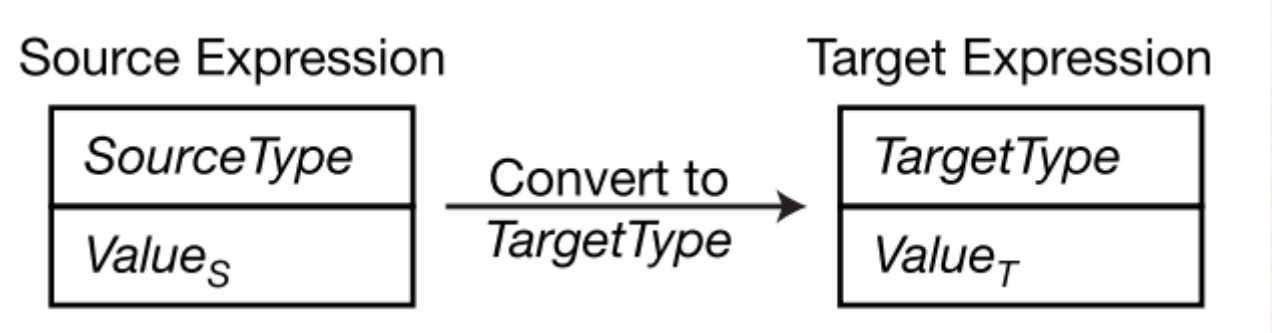
การเขียนโปรแกรม ด้วยภาษา C#

Conversions

Conversions

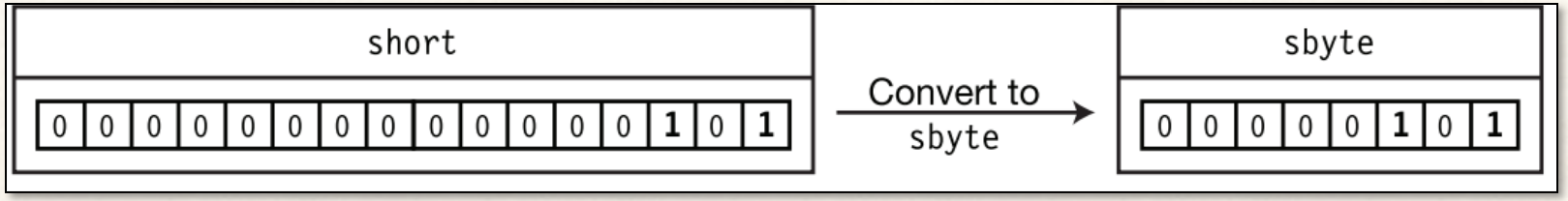
Conversions

- *เป็นกระบวนการสำหรับแปลงข้อมูลชนิดหนึ่ง (ที่ต้นทาง) ให้ดูเสมือนว่าเป็นข้อมูลอีกชนิดหนึ่ง (ที่ปลายทาง)*
- *ถ้าต้นทางและปลายทาง เป็นชนิดข้อมูลเดียวกัน ก็ไม่ต้องผ่านกระบวนการแปลงชนิด*

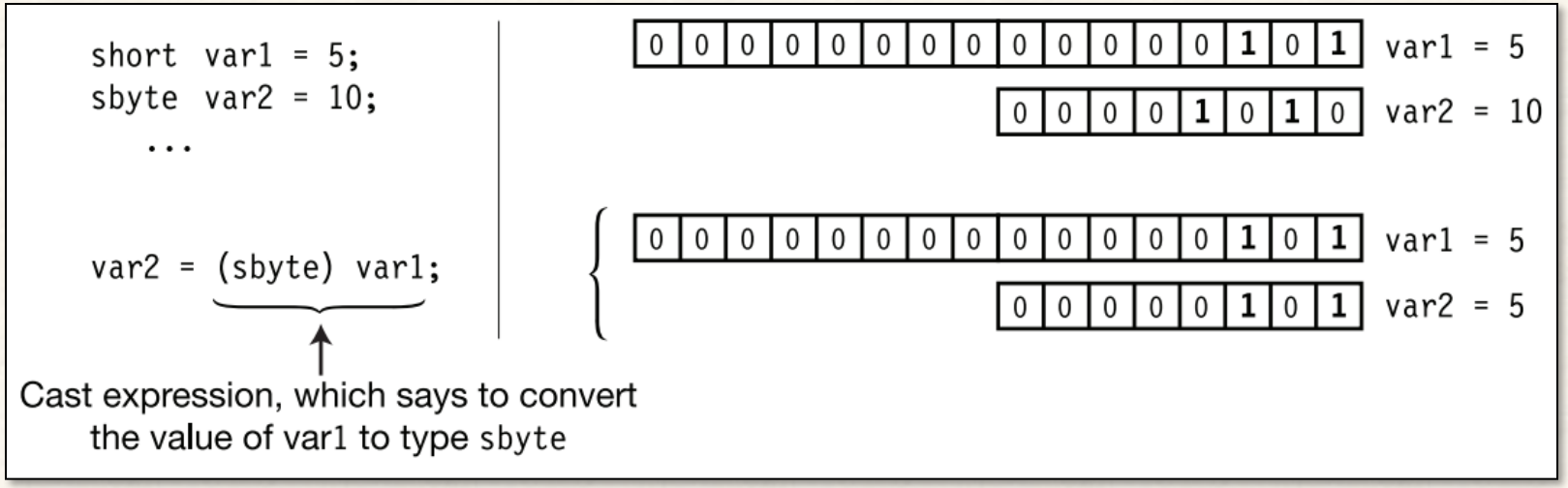


ตัวอย่างการแปลงชนิดข้อมูล

```
short var1 = 5;
sbyte var2;
Var2 = var1;
```



Cast expression

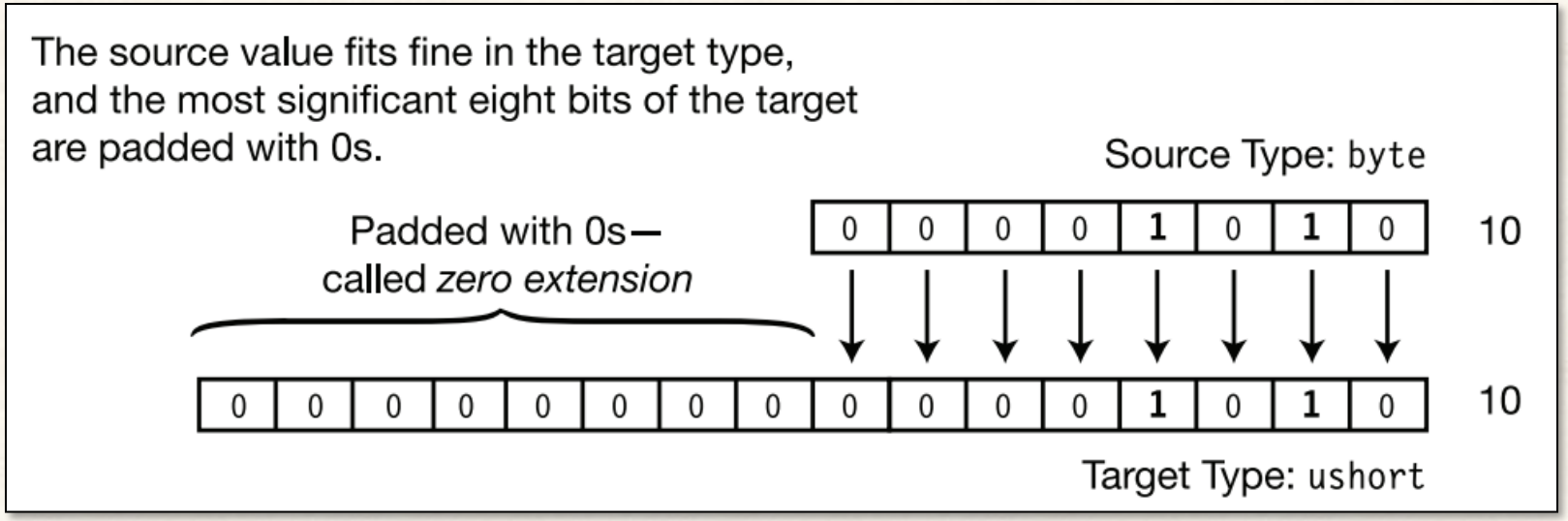


การแปลงโดยปริยาย (Implicit Conversions)

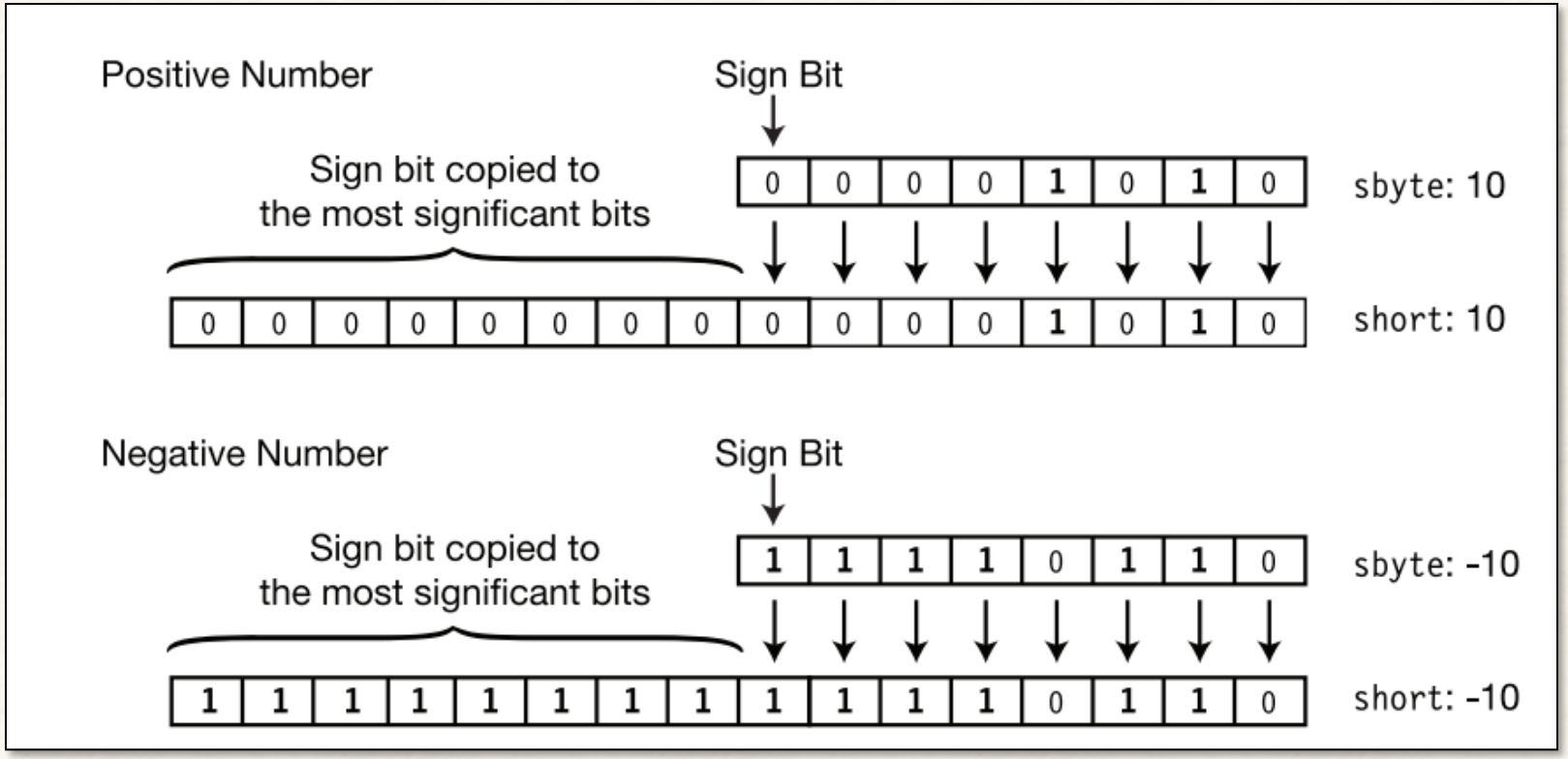
- การแปลงโดยปริยาย เป็นการแปลงที่ ตัวแปลภาษา ทำโดยอัตโนมัติ
- เมื่อตัวแปรปลายทาง มีขนาดใหญ่กว่า (มีจำนวนบิตมากกว่า) ก็จะมีการเติมตัวเลข 0 หรือ 1 ลงไปในบริเวณที่เหลือ
- ถ้าเติมเลข 0 ลงไป เรียกว่าการทำ *zero extension*

zero extension

zero extension สำหรับ unsigned conversions

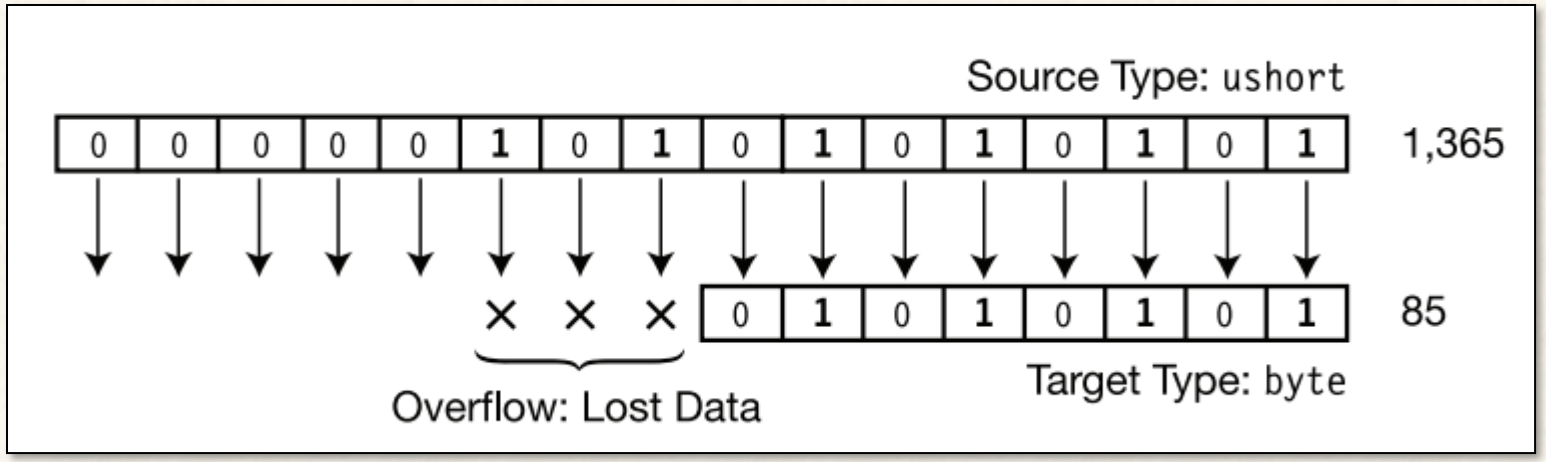


Sign extension in signed conversions



Explicit Conversions and Casting

- ushort สามารถเก็บข้อมูลในช่วง 0 และ 65,535
- byte สามารถเก็บข้อมูลในช่วง 0 และ 255
- จะเป็นอย่างไรถ้านำค่าที่เก็บใน ushort มาใส่ใน byte?

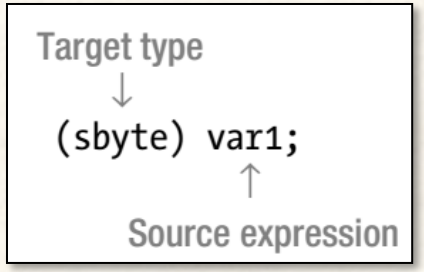


$$\frac{255}{65535} = 0.00389 \approx 0.4\%$$

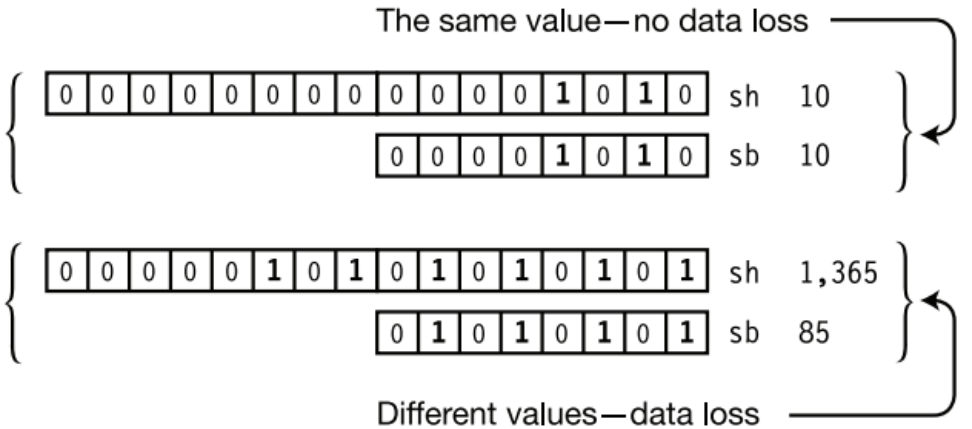
⬅ แปลงได้โดยปลอดภัย

Casting

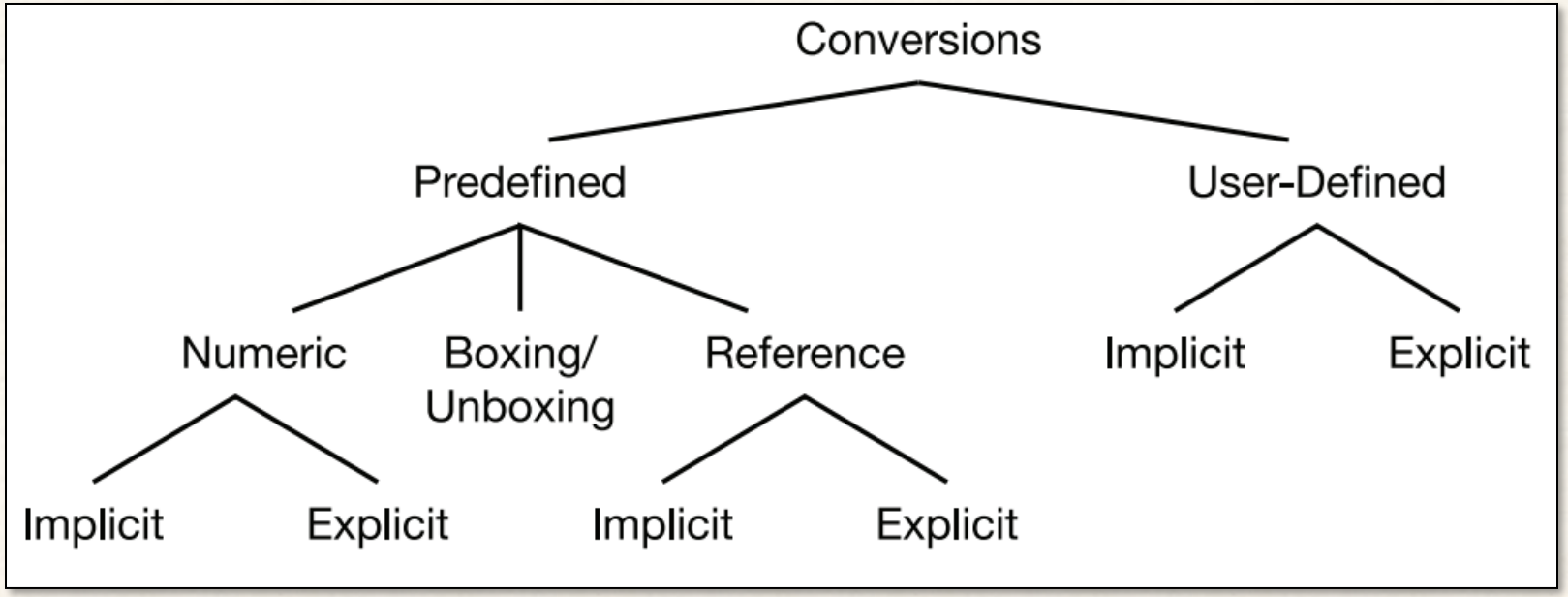
- ทำได้โดยการระบุชนิดของตัวแปรปลายทาง ไว้ในวงเล็บ หน้าตัวแปรต้นทาง



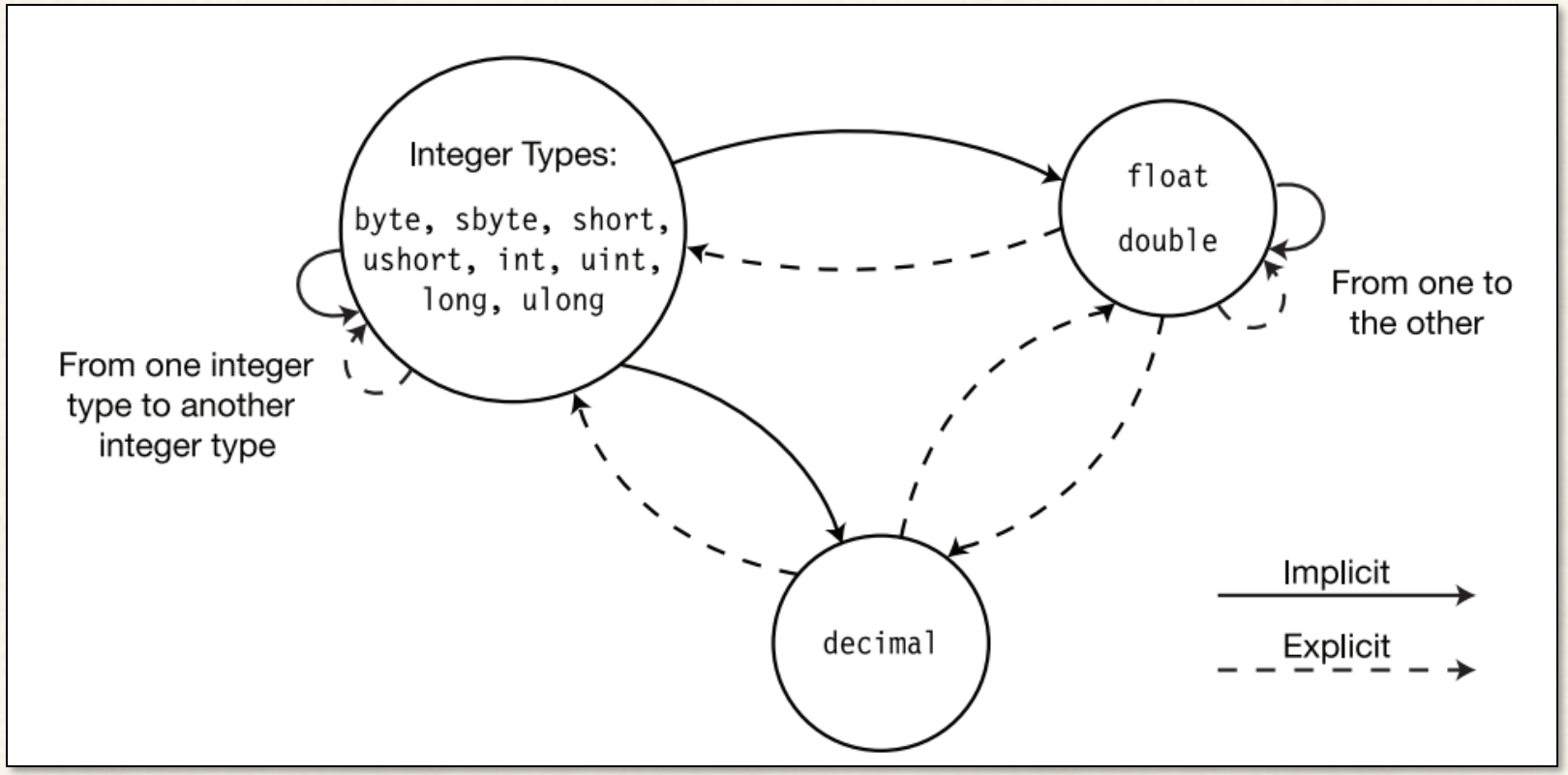

```
ushort sh = 10;
byte sb = (byte) sh;
Console.WriteLine
     ("sb:  {0} = 0x{0:X}", sb);

sh = 1365;
sb = (byte) sh;
Console.WriteLine
     ("sb:  {0} = 0x{0:X}", sb);
```



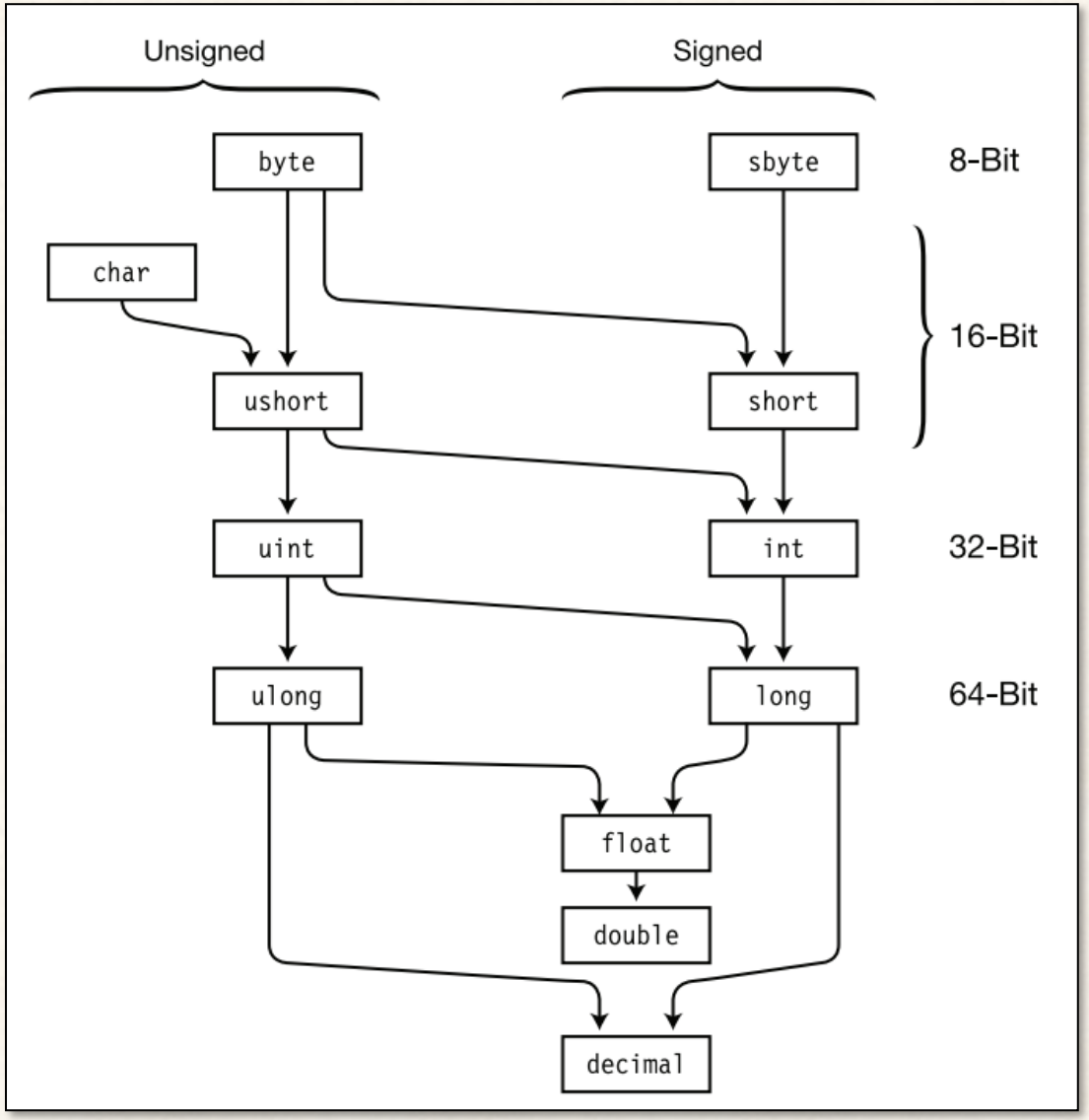
Types of Conversions



Numeric Conversions



Implicit Numeric Conversions



The checked and unchecked Operators

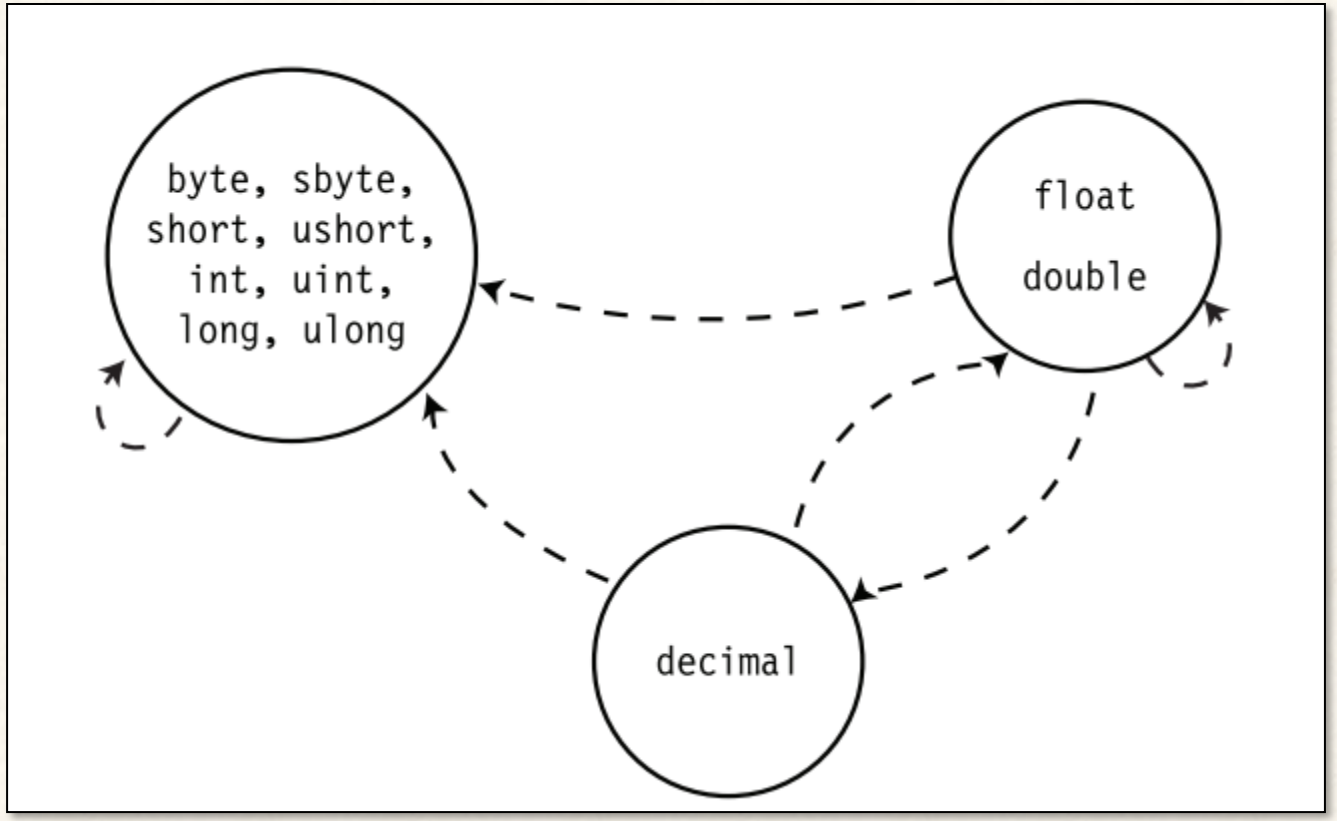
```
checked   ( Expression )
unchecked ( Expression )
```

```
ushort sh = 2000;
byte   sb;

sb = unchecked ( (byte) sh );          // Most significant bits lost
Console.WriteLine("sb: {0}", sb);

sb =   checked ( (byte) sh );          // OverflowException raised
Console.WriteLine("sb: {0}", sb);
```

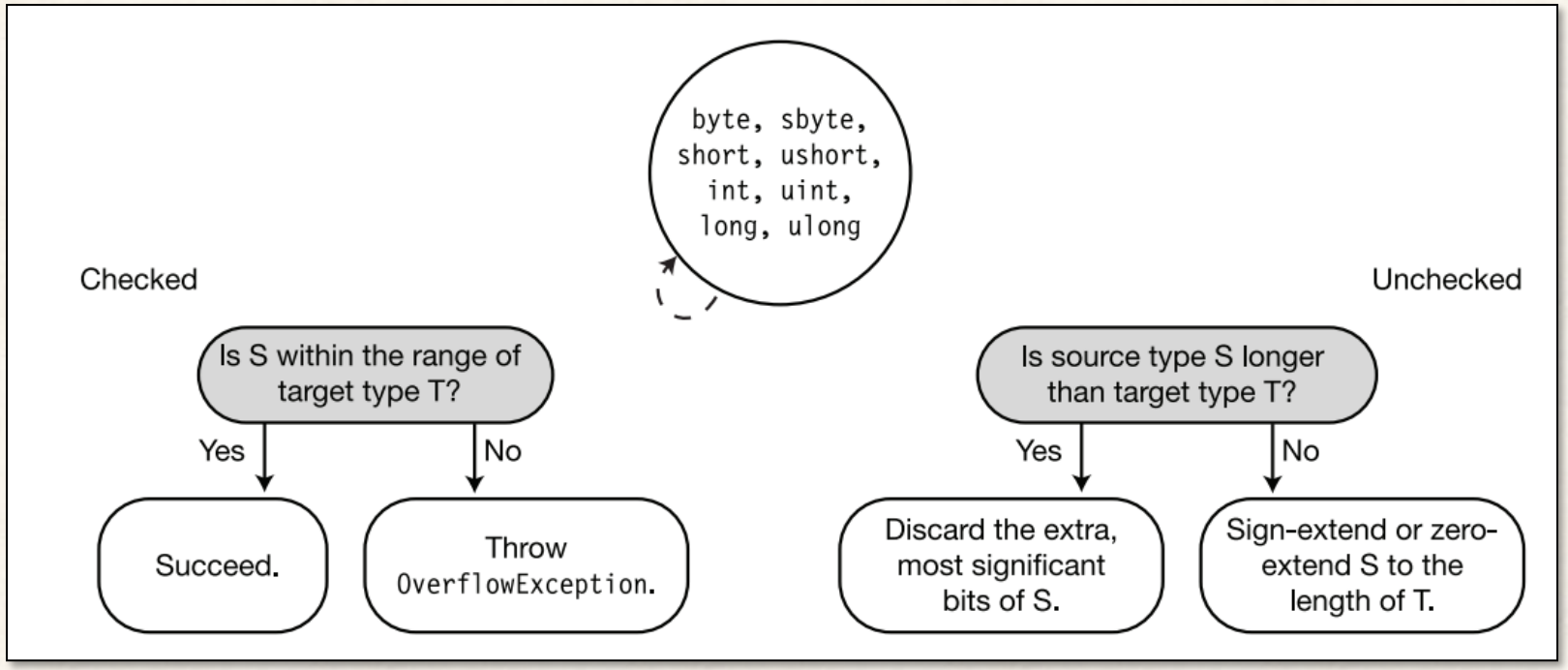
The checked and unchecked Statements

```
byte   sb;
ushort sh = 2000;

unchecked                                          // Set unchecked
{
   sb = (byte) sh;
   Console.WriteLine("sb: {0}", sb);

   checked                                         // Set checked
   {
      sb = (byte) sh;
      Console.WriteLine("sb: {0}", sh);
   }
}
```

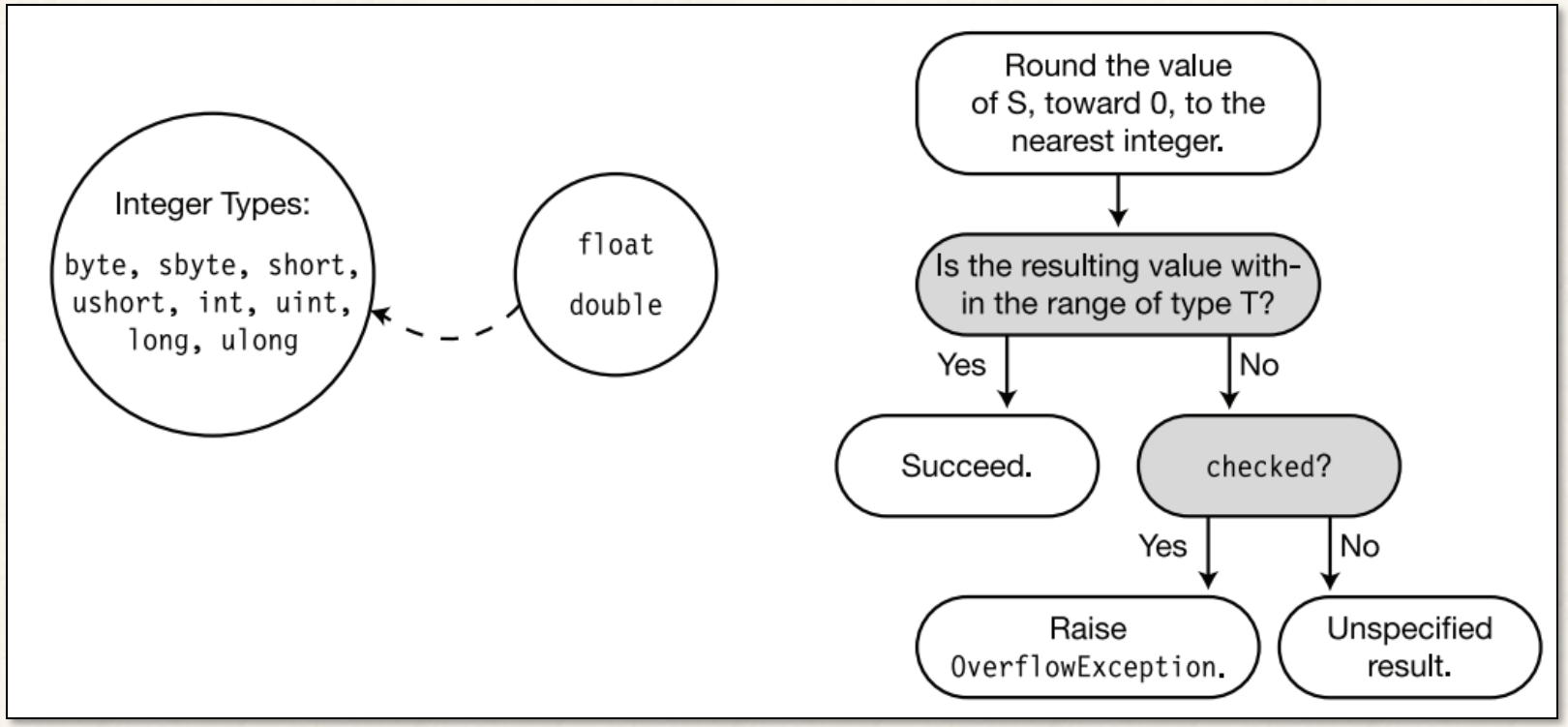
Explicit Numeric Conversions



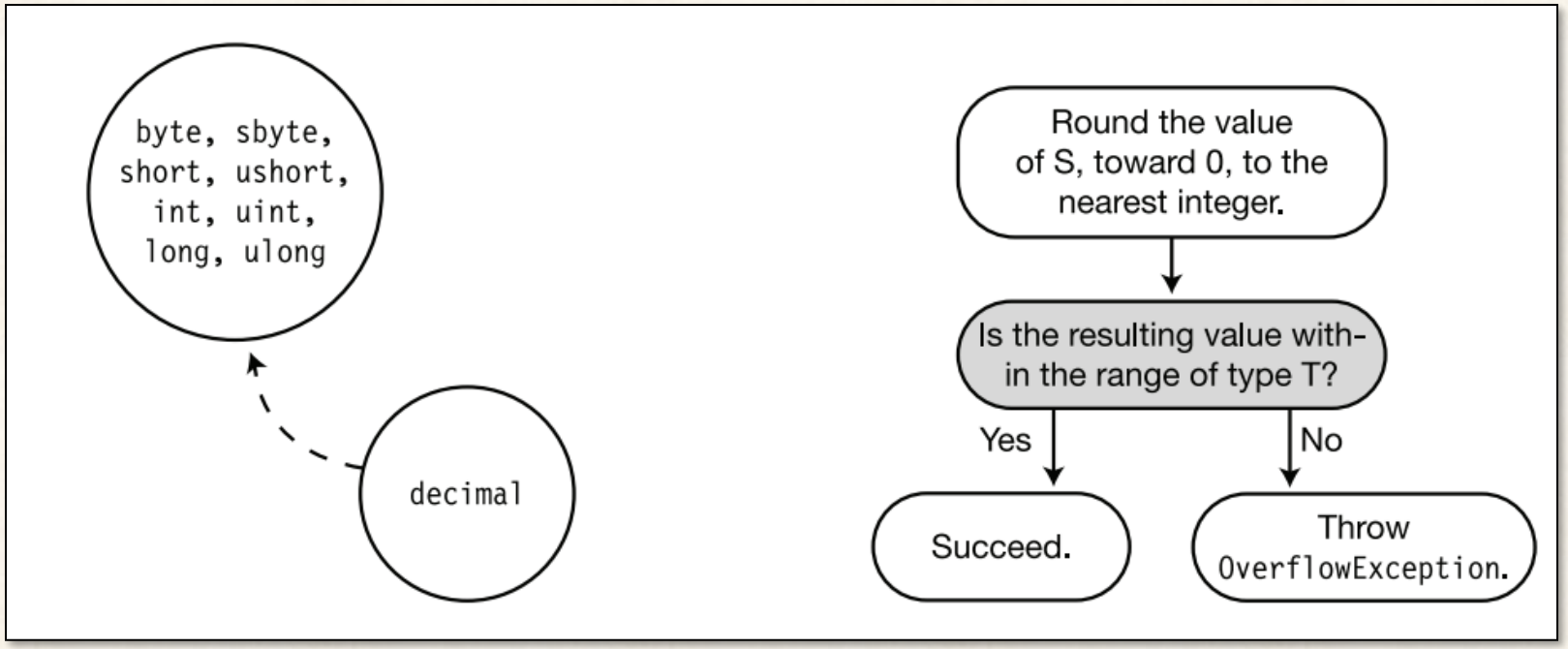
Integer Type to Integer Type



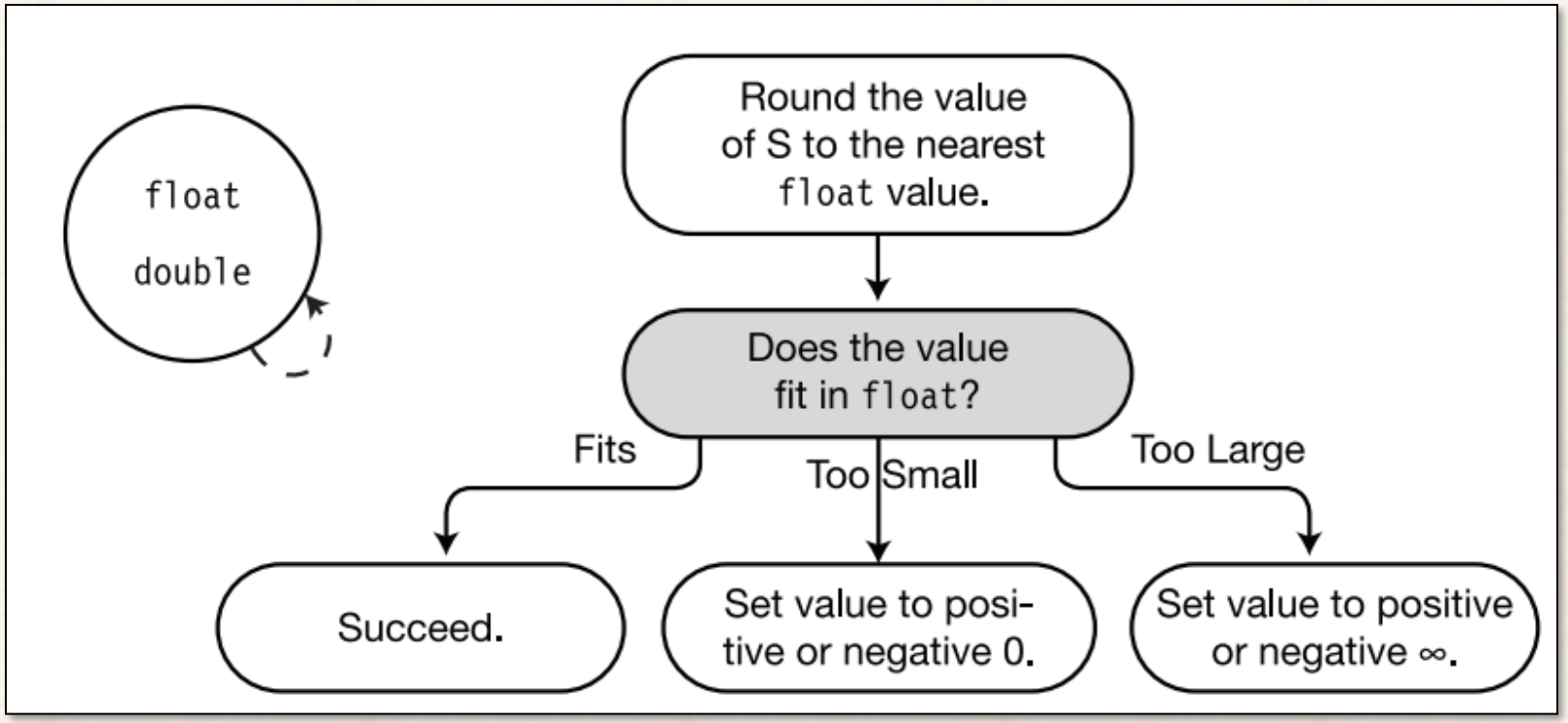
float or double to Integer Type



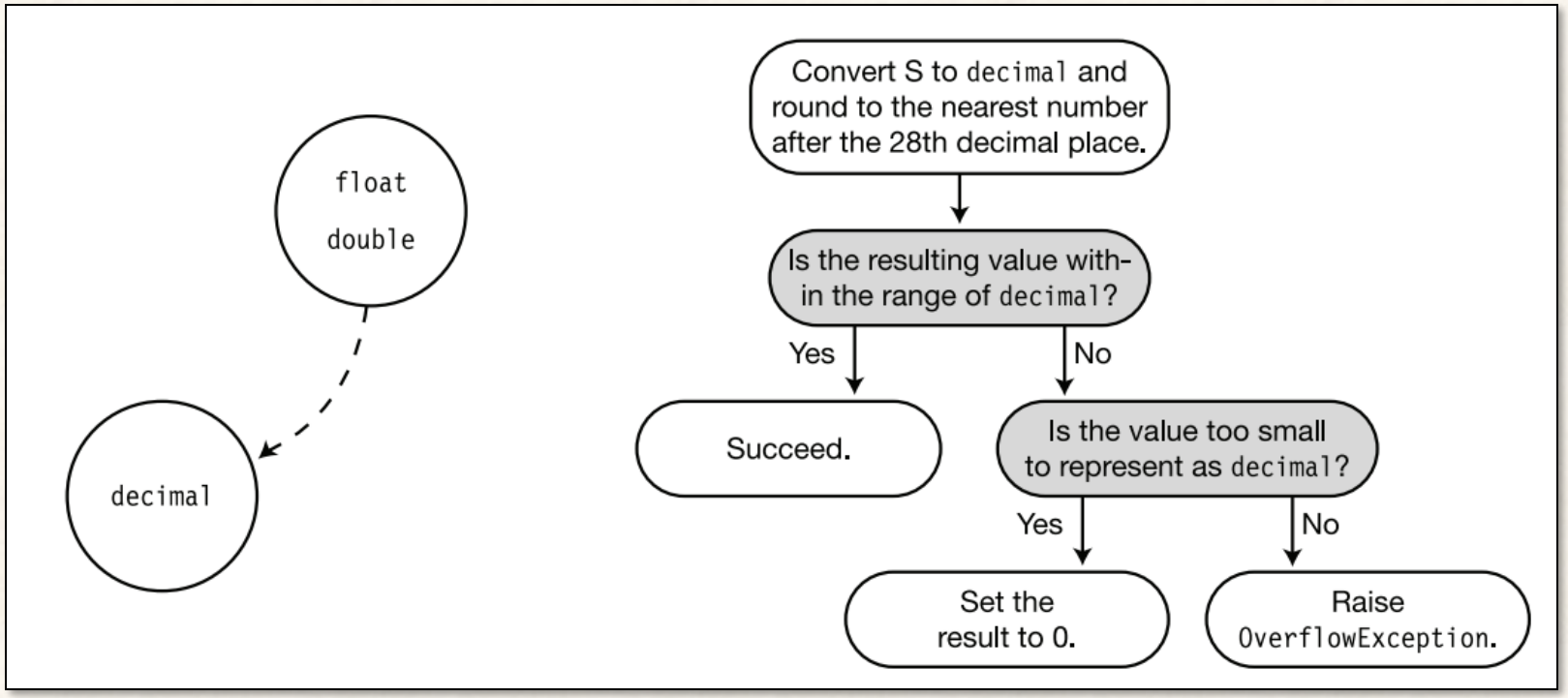
decimal to Integer Type



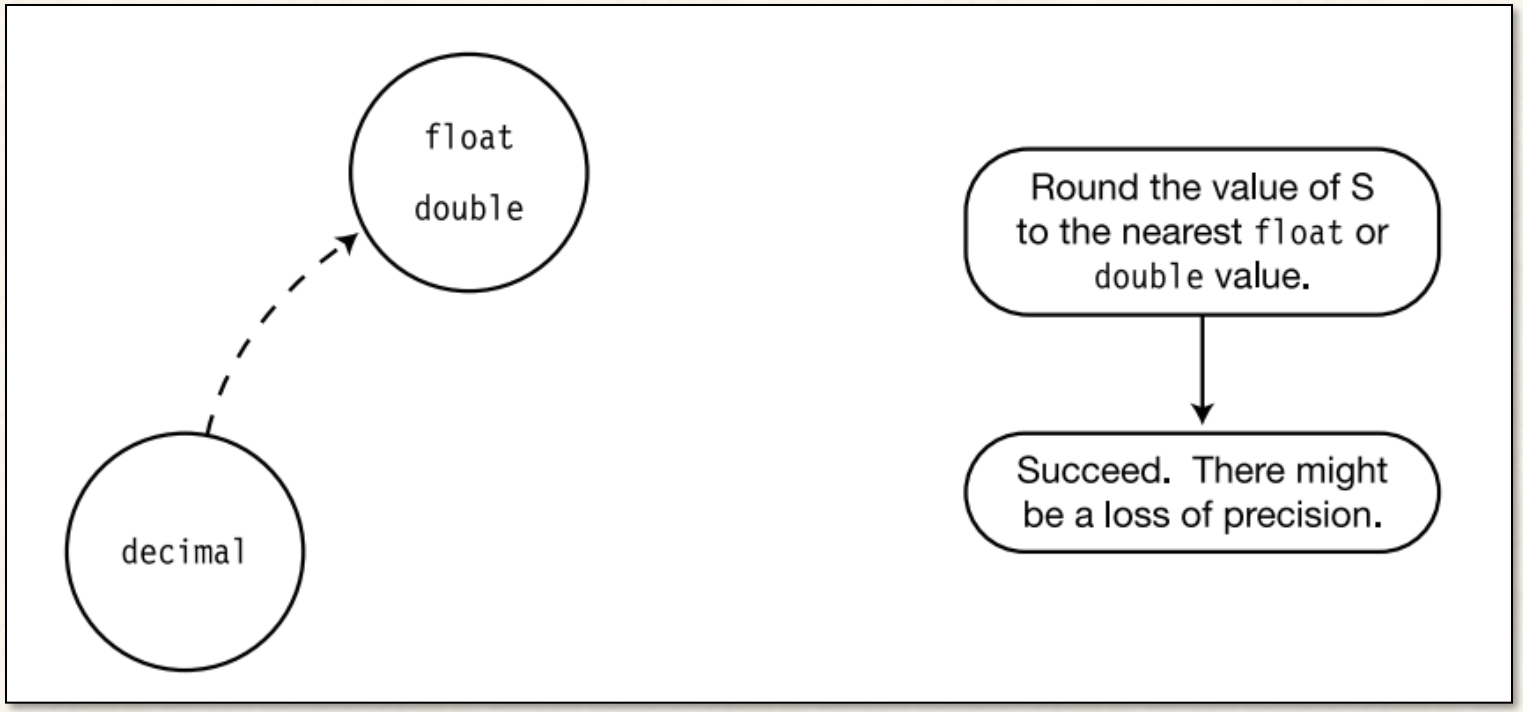
double to float



float or double to decimal



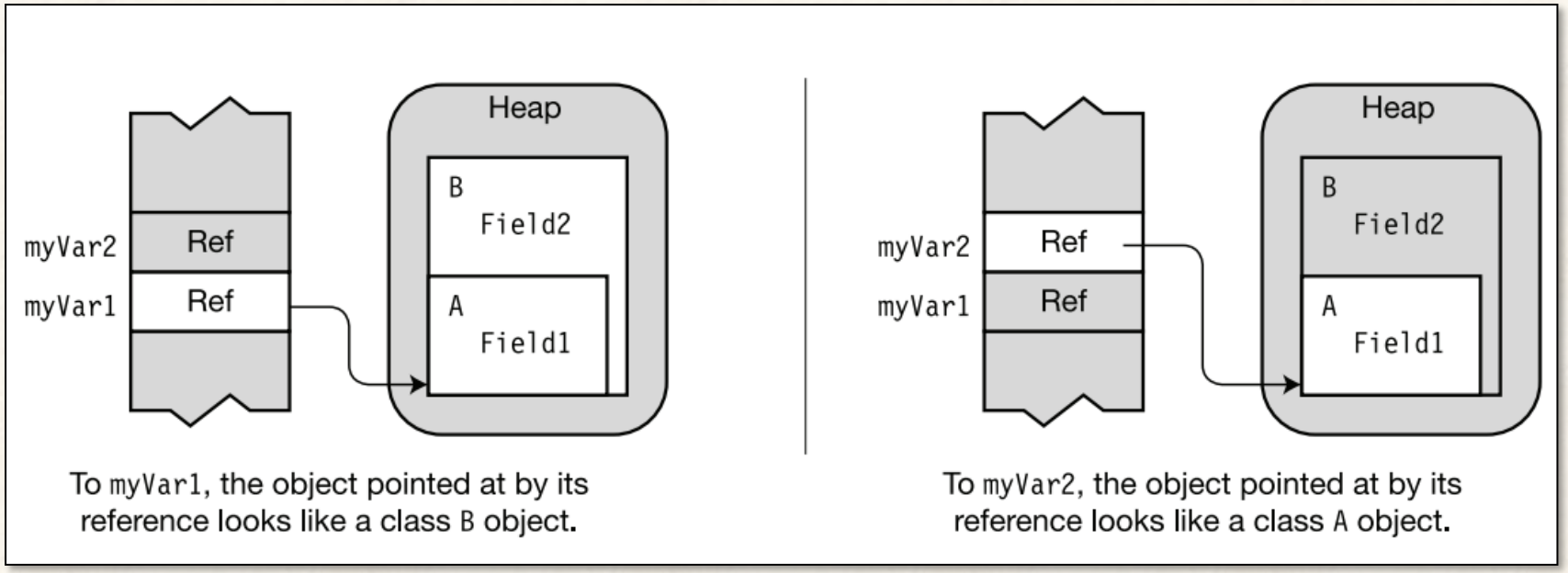
decimal to float or double



Reference Conversions

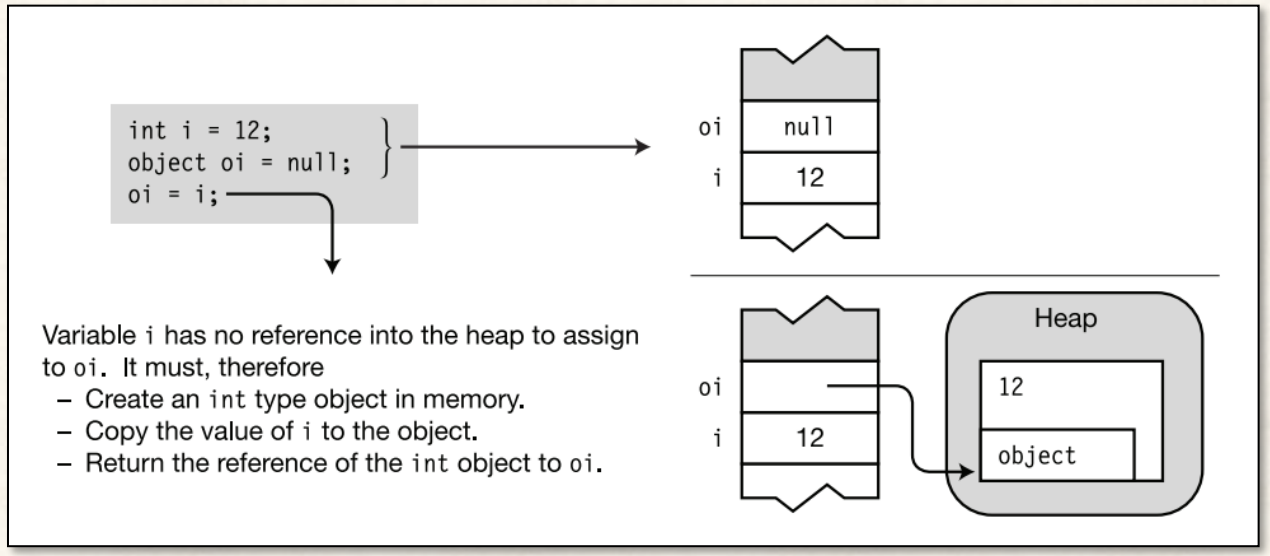
- ตัวแปร Reference คือ ตัวแปรที่มีองค์ประกอบ 2 ส่วน โดยส่วนแรก เป็นตัวแปรบน stack และส่วนที่สองเป็นวัตถุที่อยู่ใน heap
- reference conversion เป็นการแปลงตัวแปรบน stack ให้เป็นชนิดอื่น ในขณะที่ยังคงชี้ไปยังวัตถุเดิมใน heap

```
class A    { public int Field1; }

class B: A { public int Field2; }

class Program
{
   static void Main( )
   {
      B myVar1 = new B();
Return the reference to myVar1 as a reference to a class A.
                    ↓
      A myVar2 = (A) myVar1;

      Console.WriteLine("{0}", myVar2.Field1);          // Fine
      Console.WriteLine("{0}", myVar2.Field2);          // Compile error!
   }                                    ↑
}                                  myVar2 can't see Field2.
```


Heap
B
Field2
A
Field1
myVar2
Ref
myVar1
Ref
To myVar1, the object pointed at by its reference looks like a class B object.
Heap
B
Field2
A
Field1
myVar2
Ref
myVar1
Ref
To myVar2, the object pointed at by its reference looks like a class A object.

Boxing Conversions

- การแปลงจาก value type ไปเป็น reference type เรียกว่า boxing conversion
- เนื่องจาก reference type จะเก็บค่าของข้อมูลใน heap ดังนั้น ในการแปลงจะมีกระบวนการดังนี้
  1. สร้าง “วัตถุ” ใน heap
  2. คัดลอกค่าตัวแปร value type ไปยัง heap
  3. ส่งที่อยู่ของ “วัตถุ” ไปเก็บในตัวแปร reference type



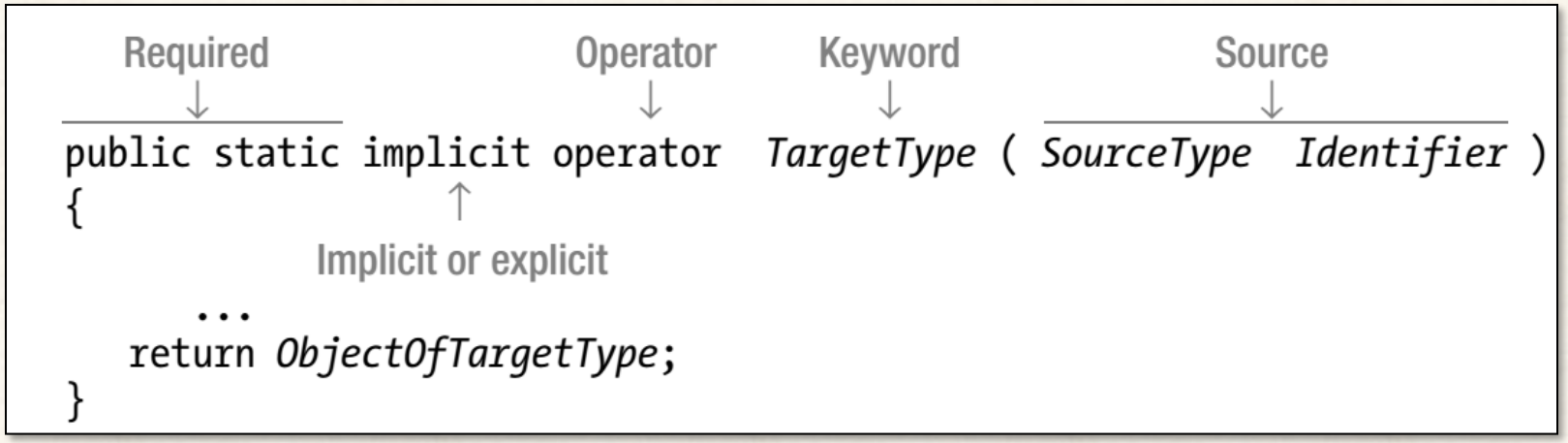
Unboxing Conversions

- เป็นกระบวนการแปลง boxed object ไปเป็น value type

```
static void Main()
{
    int i = 10;
      Box i and assign its reference to oi.
            ↓
    object oi = i;
        Unbox oi and assign its value to j.
              ↓
    int j = (int) oi;
    Console.WriteLine("i: {0},   oi: {1},   j: {2}", i,  oi, j);
}
```

User-Defined Conversions

รูปแบบ



ตัวอย่างการใช้

```
public static implicit operator int(Person p)
{
    return p.Age;
}
```

Example of a User-Defined Conversion

แบบปริยาย (Implicit)

```
class Person
{
   public string Name;
   public int    Age;
   public Person(string name, int age)
   {
      Name = name;
      Age = age;
   }

   public static implicit operator int(Person p)   // Convert Person to int.
   {
      return p.Age;
   }

   public static implicit operator Person(int i)   // Convert int to Person.
   {
      return new Person("Nemo", i);        // ("Nemo" is Latin for "No one".)
   }
}
```

```
class Program
{
   static void Main( )
   {
      Person bill = new Person( "bill", 25);

      int age = bill;
      Console.WriteLine("Person Info: {0}, {1}", bill.Name, age);

      Person anon = 35;
      Console.WriteLine("Person Info: {0}, {1}", anon.Name, anon.Age);
   }
}
```

Convert a Person object to an int.

Convert an int to a Person object.

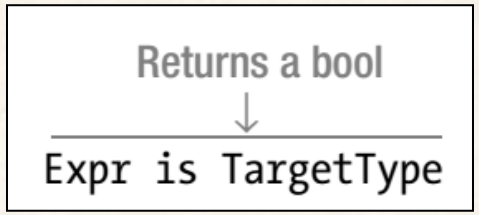
Example of a User-Defined Conversion

แบบแจ้งชัด (Explicit)

```
                    Explicit
                       ↓
        ...
public static explicit operator int( Person p )
{
   return p.Age;
}

...

static void Main( )
{
        ...   Requires cast expression
                    ↓
   int age = (int) bill;
        ...
```

The is Operator

- เนื่องจากในบางครั้ง เราไม่แน่ใจว่า การ conversion จะประสบผลสำเร็จหรือไม่ ซึ่งการ conversion ที่ไม่สำเร็จ อาจทำให้เกิด exception ได้
- ทางออกคือการใช้ is operator



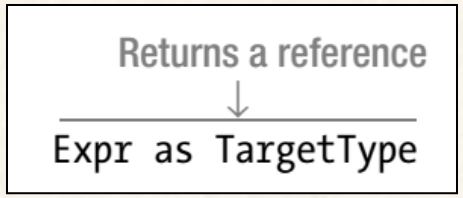
- is operator ใช้ได้กับ
  - reference conversion
  - boxing conversion
  - unboxing conversion

ตัวอย่าง is operator

```
class Employee : Person { }
class Person
{
   public string Name = "Anonymous";
   public int Age     = 25;
}

class Program
{
   static void Main()
   {
      Employee bill = new Employee();
      Person p;

      // Check if variable bill can be converted to type Person
      if( bill is Person )
      {
         p = bill;
         Console.WriteLine("Person Info: {0}, {1}", p.Name, p.Age);
      }
   }
}
```

The as Operator

- ใช้งานคล้ายกับ cast operator แต่จะมีกลไกป้องกันไม่ให้เกิด exception
- ถ้าไม่สามารถแปลงได้ จะส่งค่า null กลับมา



ตัวอย่าง as Operator

```
class Employee : Person { }

class Person
{
   public string Name = "Anonymous";
   public int Age     = 25;
}

class Program
{
   static void Main()
   {
      Employee bill = new Employee();
      Person p;

      p = bill as Person;
      if( p != null )
      {
         Console.WriteLine("Person Info: {0}, {1}", p.Name, p.Age);
      }
   }
}
```

ข้อจำกัดของ is และ as operator

- ใช้ได้กับ
  - Reference conversion
  - Boxing conversion
  - Unboxing conversion
- ใช้ไม่ได้กับ user-defined conversion