Domaines Discrets

Guillaume Damiand

Laboratoire d'InfoRmatique en Image et Systèmes d'information
LIRIS UMR 5205 CNRS
Université Claude Bernard, Bâtiment Nautibus (710),
43, Boulevard du 11 Novembre 1918, 69622 Villeurbanne Cedex
http://liris.cnrs.fr

29 August 2011

HyperRectDomain

- templated by a SpaceND
- constructed from two points

```
typedef SpaceND<3> TSpace;
TSpace::Point a(1, 1, 1);
TSpace::Point b(5, 5, 5);
typedef HyperRectDomain<TSpace> TDomain;
TDomain domain(a,b);
```

Iterate through HyperRectDomain

thanks to `Range` concept

- types `ConstIterator` and `ReverseConstIterator`
- methods `begin()` and `end()`
- methods `rbegin()` and `rend()`

`ConstRange` class

`ConstRange` : range through the whole domain
Inner class of HyperRectDomain

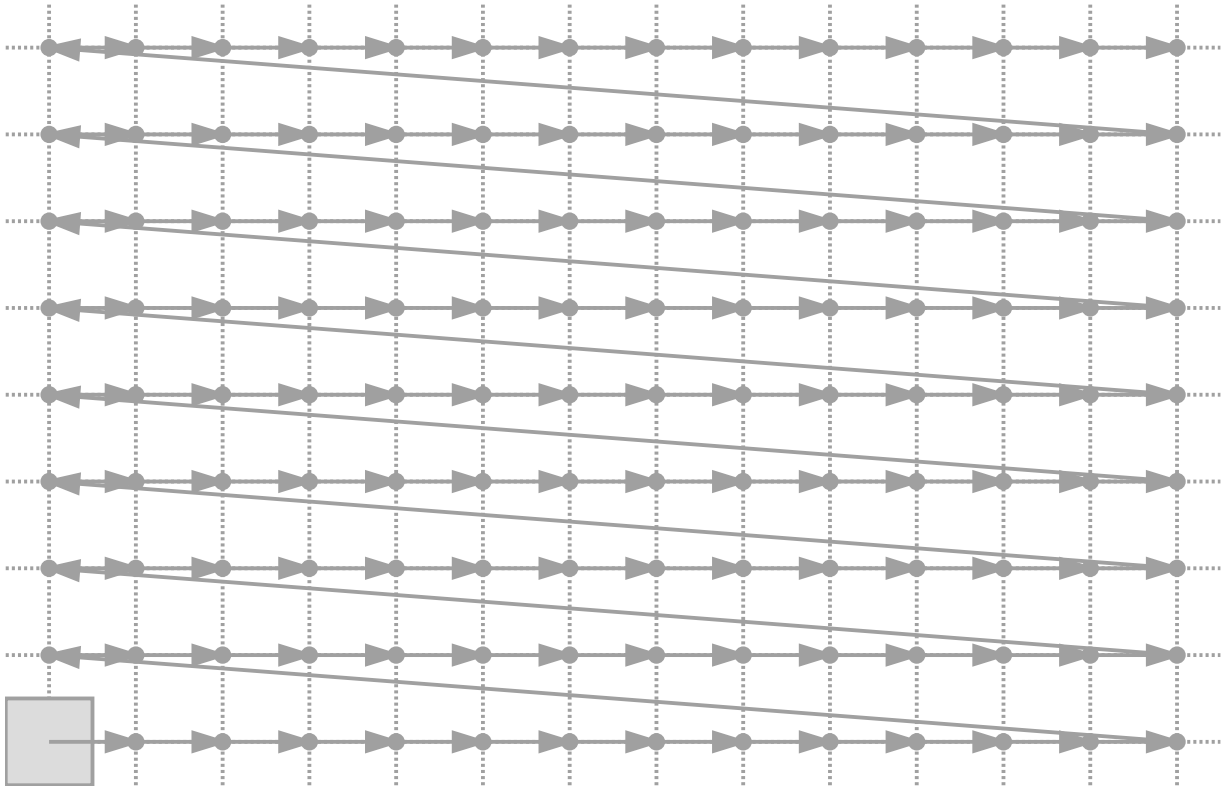
```
struct ConstRange
  {
    ConstRange(const HyperRectDomain<TSpace>& adomain);
    const ConstIterator& begin() const;
    const ConstIterator& end() const;
    ConstIterator begin(const Point& aPoint) const;
  };
const ConstRange& range() const;
```

idem for `rbegin/rend` and `ReverseConstIterator`

Example 1

```
typedef HyperRectDomain<Space> MyDomain;
Point a(-3,-4);
Point b(10,4);
MyDomain domain(a,b);
for( MyDomain::ConstIterator it = domain.range().begin(),
    itend = domain.range().end();
    it != itend; ++it)
  ...
```

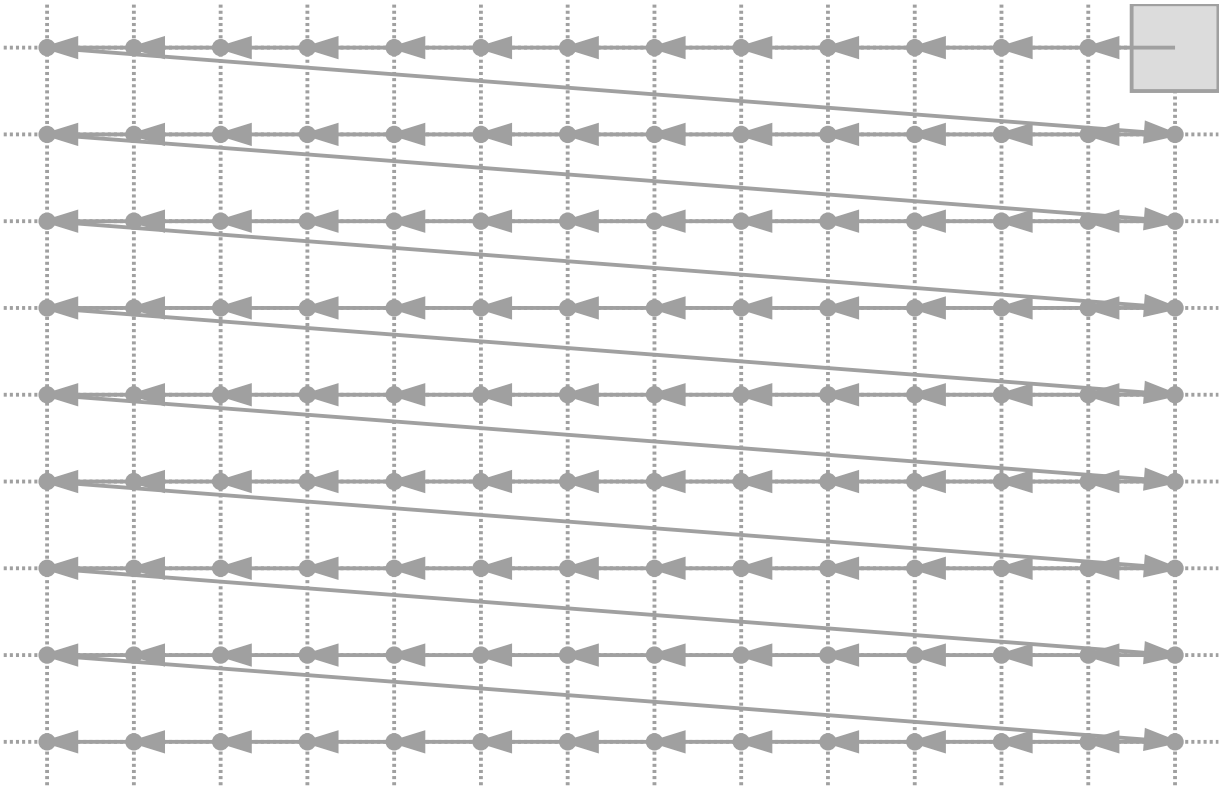
Example 1



Example 2

```
typedef HyperRectDomain<Space> MyDomain;
Point a(-3,-4);
Point b(10,4);
MyDomain domain(a,b);
for( MyDomain::ReverseConstIterator it = domain.range().
    rbegin(),
    itend = domain.range().rend();
    it != itend; ++it)
  ...
```

Example 2



`ConstSubRange` class

`ConstRange` : range through a sub domain
Inner class of HyperRectDomain

```
struct ConstSubRange
  {
    ConstSubRange(const HyperRectDomain<TSpace>& domain,
                  const std::vector<Dimension> & permutation,
                  const Point & startingPoint);
    ConstIterator begin() const;
    ConstIterator end() const;
    ConstIterator begin(const Point& aPoint) const;
  };
ConstSubRange subRange(const std::vector<Dimension> &
                       permutation) const;
ConstSubRange subRange(const std::vector<Dimension> &
                        permutation,
                        const Point & startingPoint) const;
```

idem for `rbegin/rend` and `ReverseConstIterator`

Example 3

```
typedef SpaceND<3> TSpace;
 TSpace::Point a(1, 1, 1);
 TSpace::Point b(5, 5, 5);
 HyperRectDomain<TSpace> domain(a,b);

 std::vector<TSpace::Dimension> v(2); v[0]=2; v[1]=1;
 TSpace::Point c(3,1,1);
 for( HyperRectDomain<TSpace>::ConstSubRange::
       ReverseConstIterator
       it = domain.subRange(v, c).rbegin(),
       itend = domain.subRange(v, c).rend();
       it != itend; ++it)
 ...
```

Viva C++0X

Same example with C++0X Initializer list

```
typedef SpaceND<3> TSpace;
TSpace::Point a(1, 1, 1);
TSpace::Point b(5, 5, 5);
HyperRectDomain<TSpace> domain(a,b);
TSpace::Point c(3,1,1);

for( HyperRectDomain<TSpace>::ConstSubRange::
       ReverseConstIterator
       it = domain.subRange({2,1}, c).rbegin(),
       itend = domain.subRange({2,1}, c).rend();
       it != itend; ++it)
 ...
```

`ConstSubRange` class

Shortcuts for 1, 2 and 3 parameters

```
ConstSubRange subRange(Dimension adim,
                       const Point & startingPoint) const;
ConstSubRange subRange(Dimension adim1,
                       Dimension adim2,
                       const Point & startingPoint) const;
ConstSubRange subRange(Dimension adim1,
                       Dimension adim2,
                       Dimension adim3,
                       const Point & startingPoint) const;
```

Conclusion

Interest of ranges :

- regroup things
- only one parameter in functions
  $\Rightarrow$simplification + avoid bug
- can factorize some computation/data members
- will be used by the `foreach` c++0x statment

Drawbacks(?) :

- add one indirection due to the inner class

Future works

- extend the concept of range :
  add `size()` and `empty()`
- wrong solution for ConstRange
  `HyperRectDomain` IS A range
  $\Rightarrow$ remove the inner class
  and keep the `begin()` and `end()` methods
- use range concept in other classes