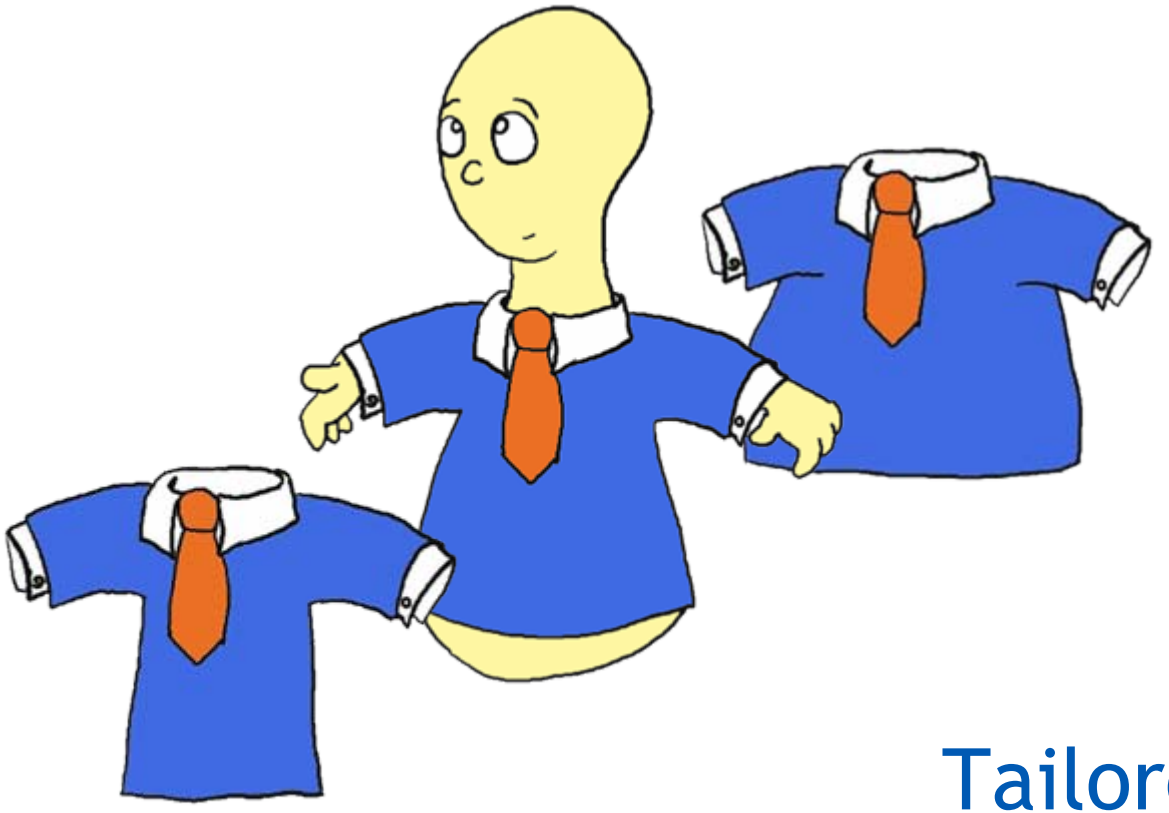
Freie Universität Berlin



Tailored off the peg

Static Polymorphism in Software Library Design

Andreas Döring

11/2007

SeqAn - Sequence Analysis

www.seqan.de

SeqAn – Sequence Analysis

- Searching in long strings
- Quite simple concepts
- Main problem: Performance!
- C++

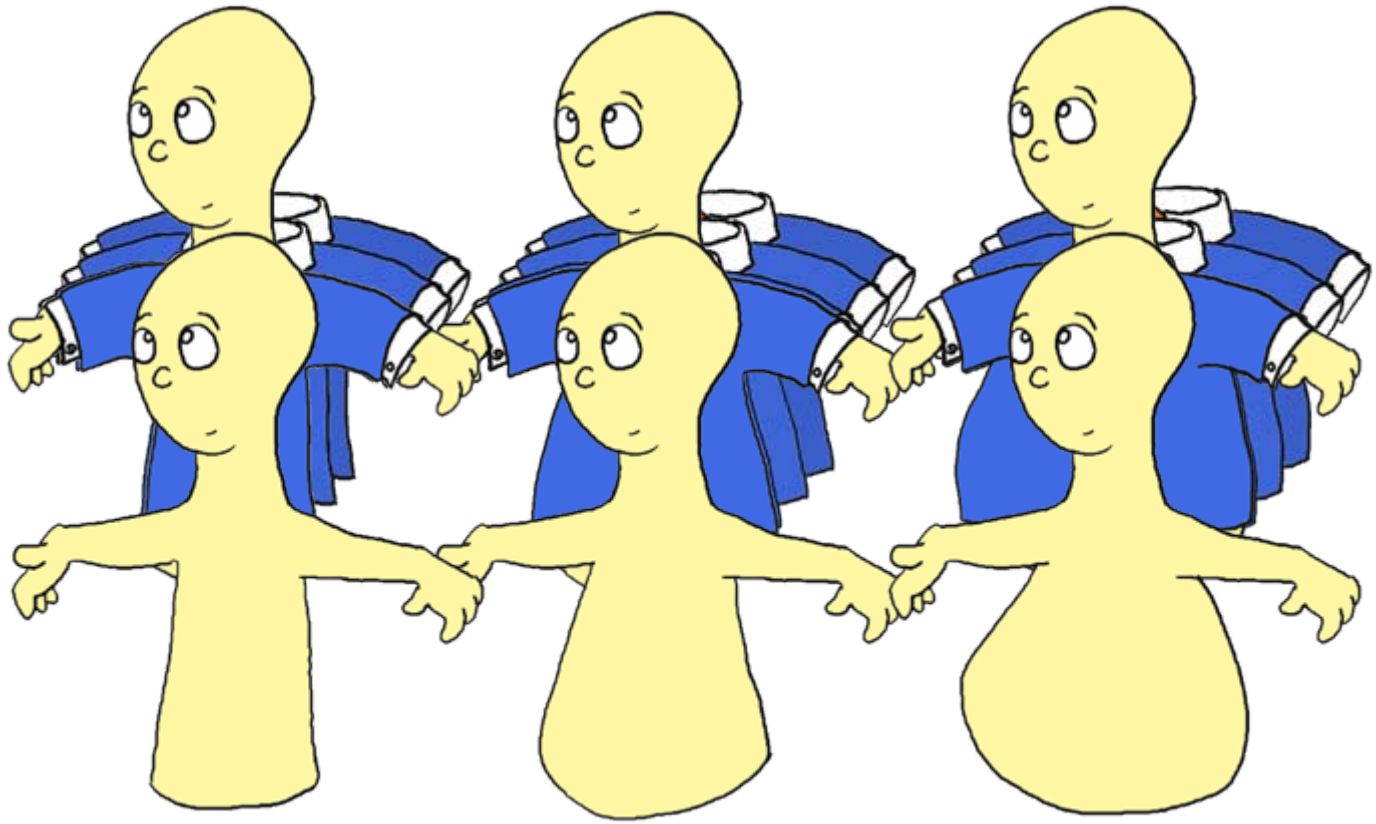
www.seqan.de

Tailored off the Peg

Speed up trick:

different implementations

for different cases

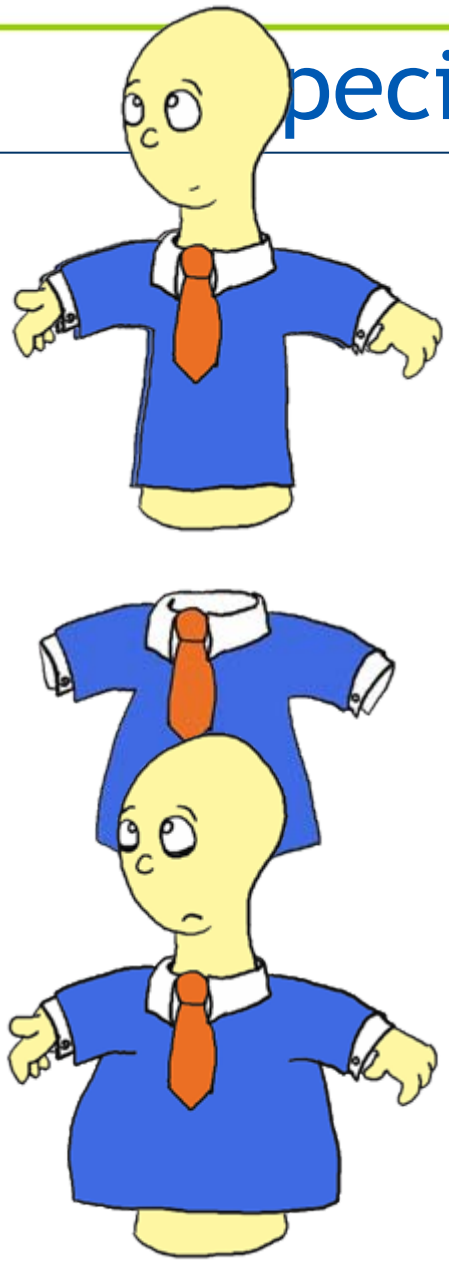


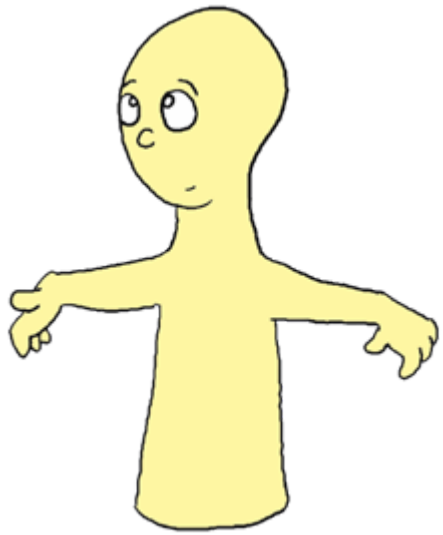
Andreas Döring: "Tailored off the peg" (2007)

pecialization Hierarchy

special — fast

general — slow





Andreas Döring: "Tailored off the peg" (2007)

Run Time "Tailoring"

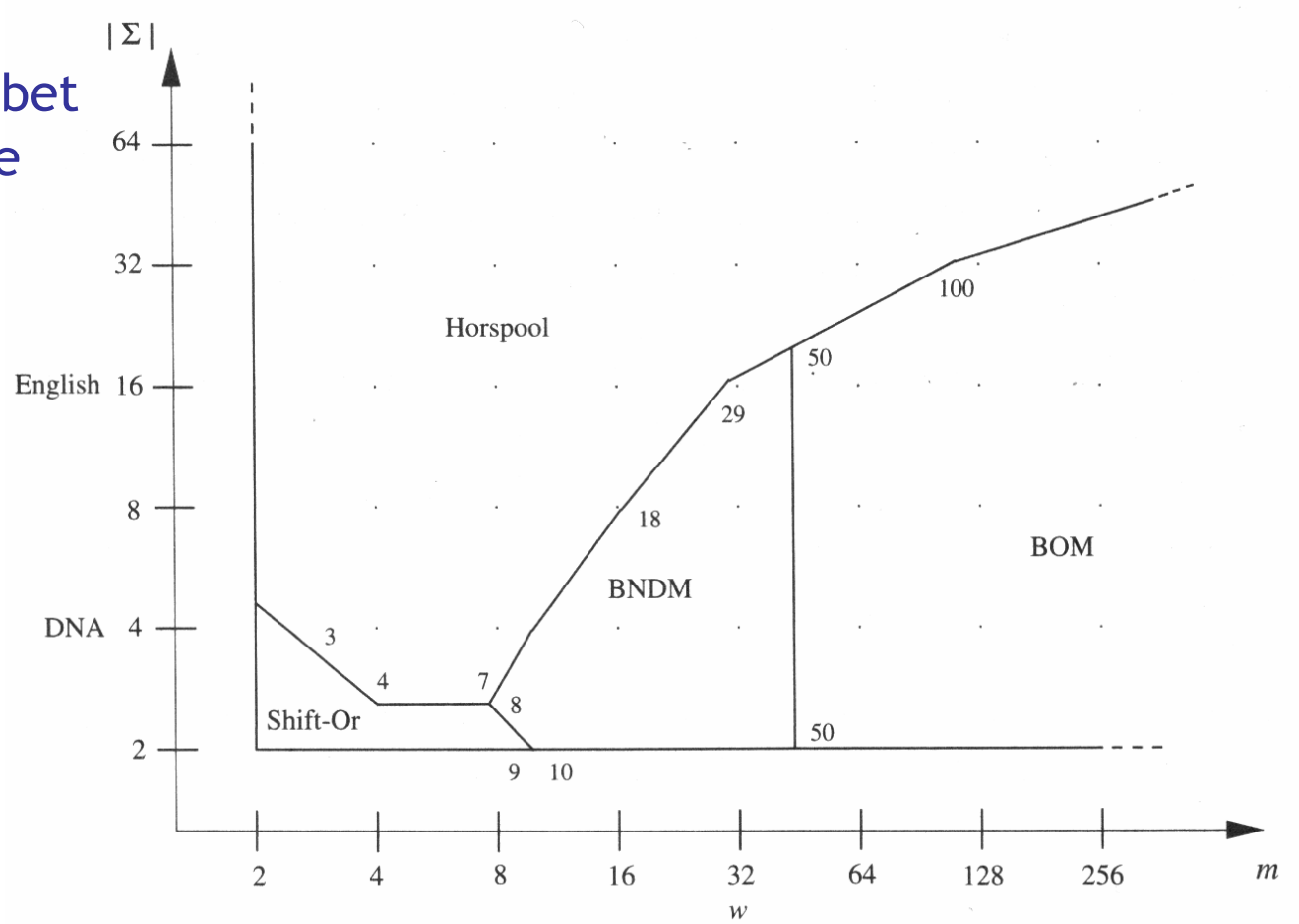
Select best variant

at run time

depending on the data

Run Time "Tailoring"

String matching algorithms:



[Navarro, Raffinot, "Flexible Pattern Matching in Strings" (2002)]

Compile Time "Tailoring"

Select best variant

at compile time

depending on data types

Gapped q-Gram Hash

Given:

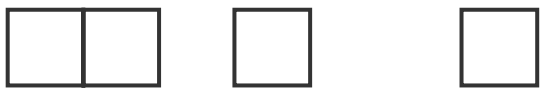
1. Alphabet:

$$\Sigma = \{A,C,G,T\}$$

2. Sequence:

A C G T G C C A T G C C A T C G T G C …

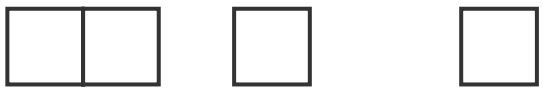
3. Shape:



Gapped q-Gram Hash

$\Sigma = \{A,C,G,T\}$

A C G T G C C A T G C C A T C G T G C …



Gapped q-Gram Hash

To do:

Compute hash values for all positions!

$\Sigma = \{A,C,G,T\}$

0 1 2 3

A C G T G C C A T G C C A T C G T G C …

$0\ 1\ 3\ 0_4 = 28_{10}$ hash value

Gapped q-Gram Hash

To do:

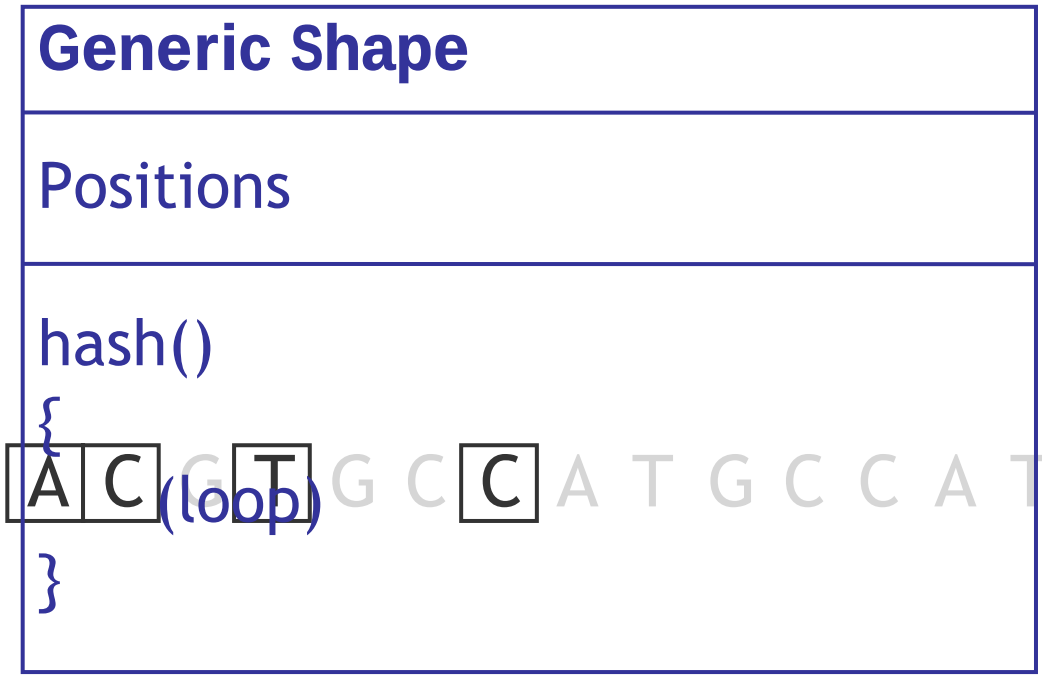
Compute hash values for all positions!

A C G T G C C A T G C C A T C G T G C …

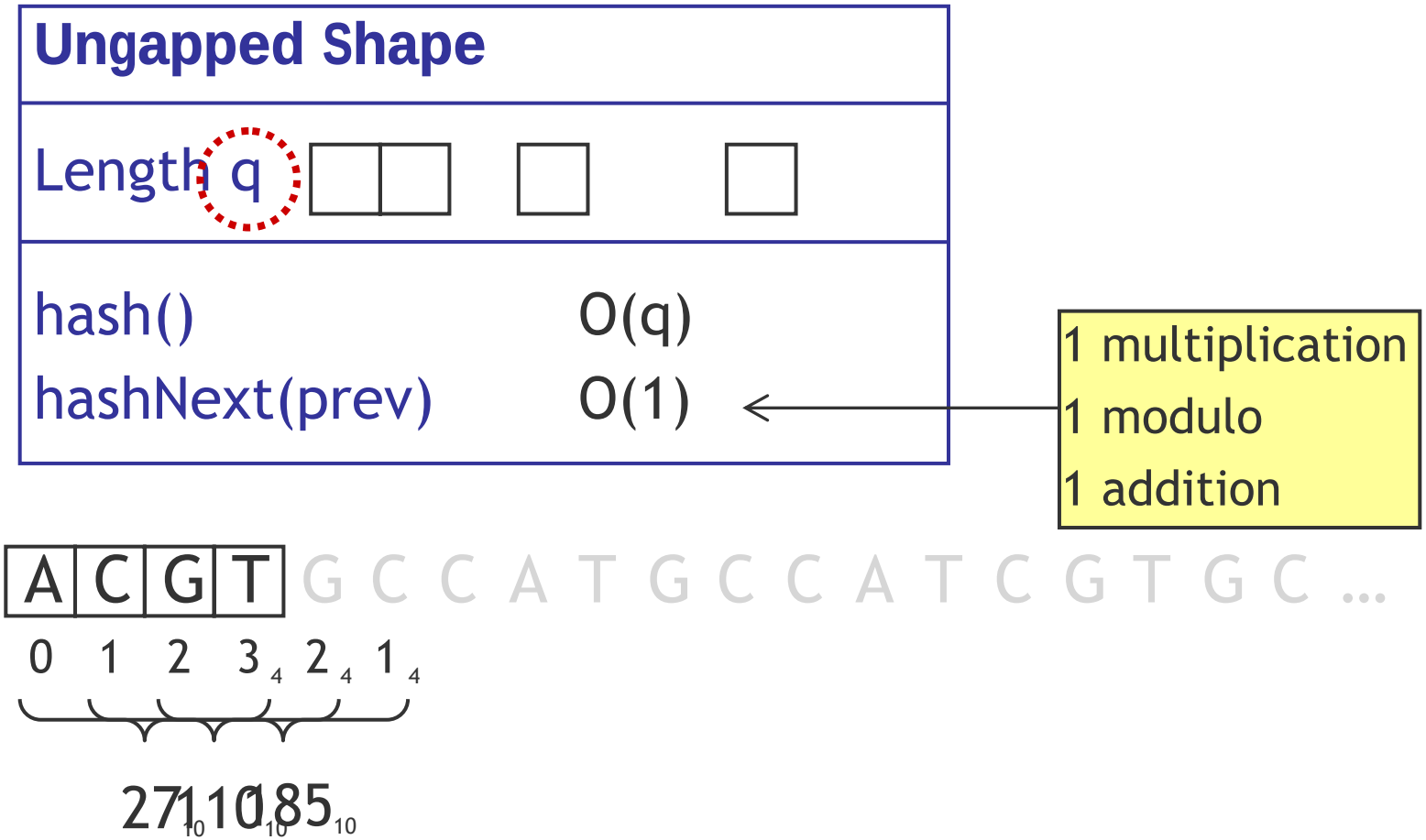
Application:

Fast scan for similar regions in two sequences

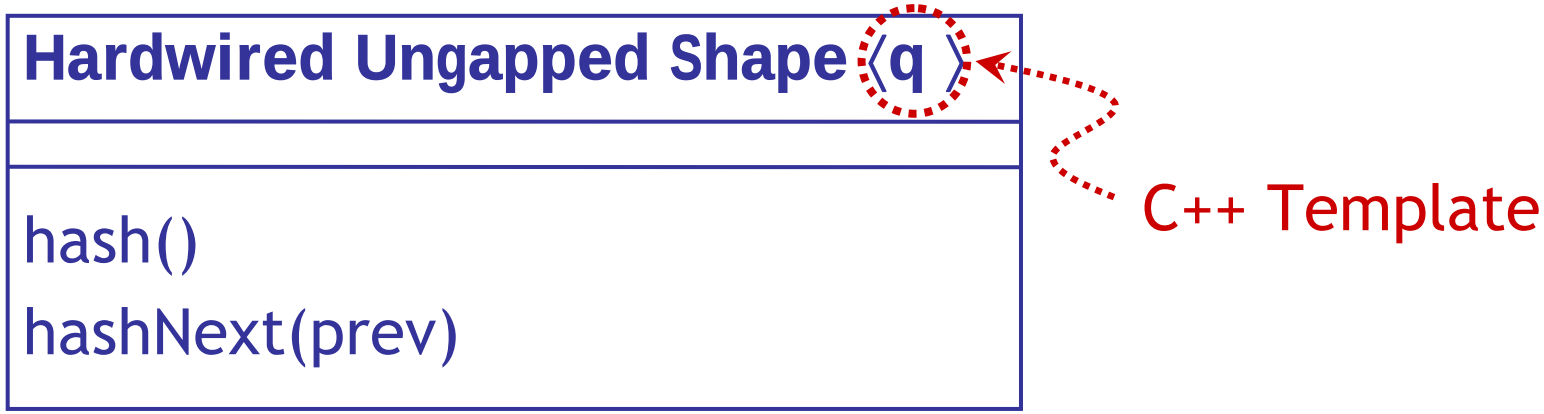
Implementations



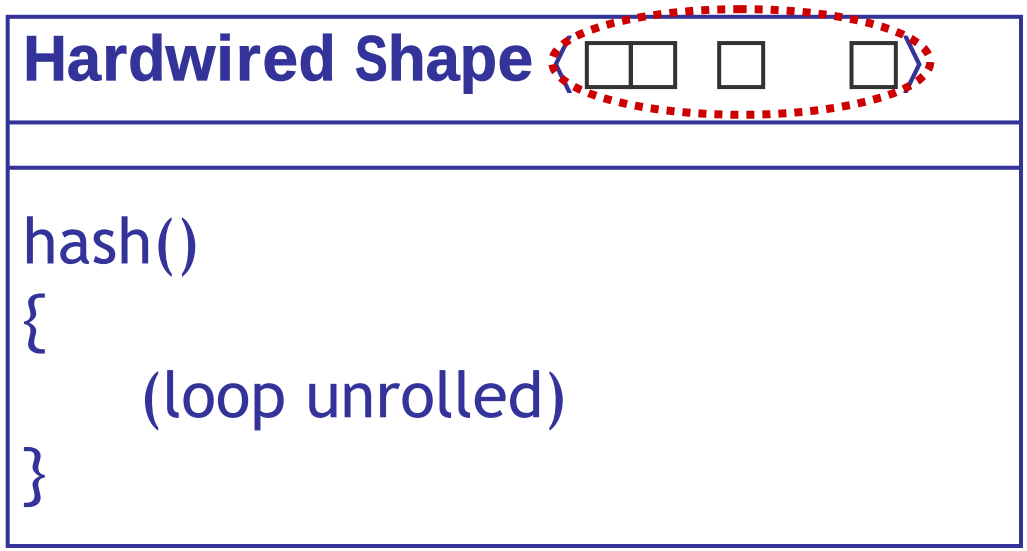
Implementations



Implementations



Implementations



Polymorphism

hashAll(Sequence, Shape)

hashAll(Sequence, GenericShape)
hashAll(Sequence, HardwiredShape ⟨...⟩)
hashAll(Sequence, UngappedShape)
hashAll(Sequence, HardwiredUngappedShape ⟨...⟩)

Generic Algorithm

```
hashAll(Sequence, Shape)
{
    for position=0 to Sequence.length():
        Shape.hash(position)
}
```

Generic

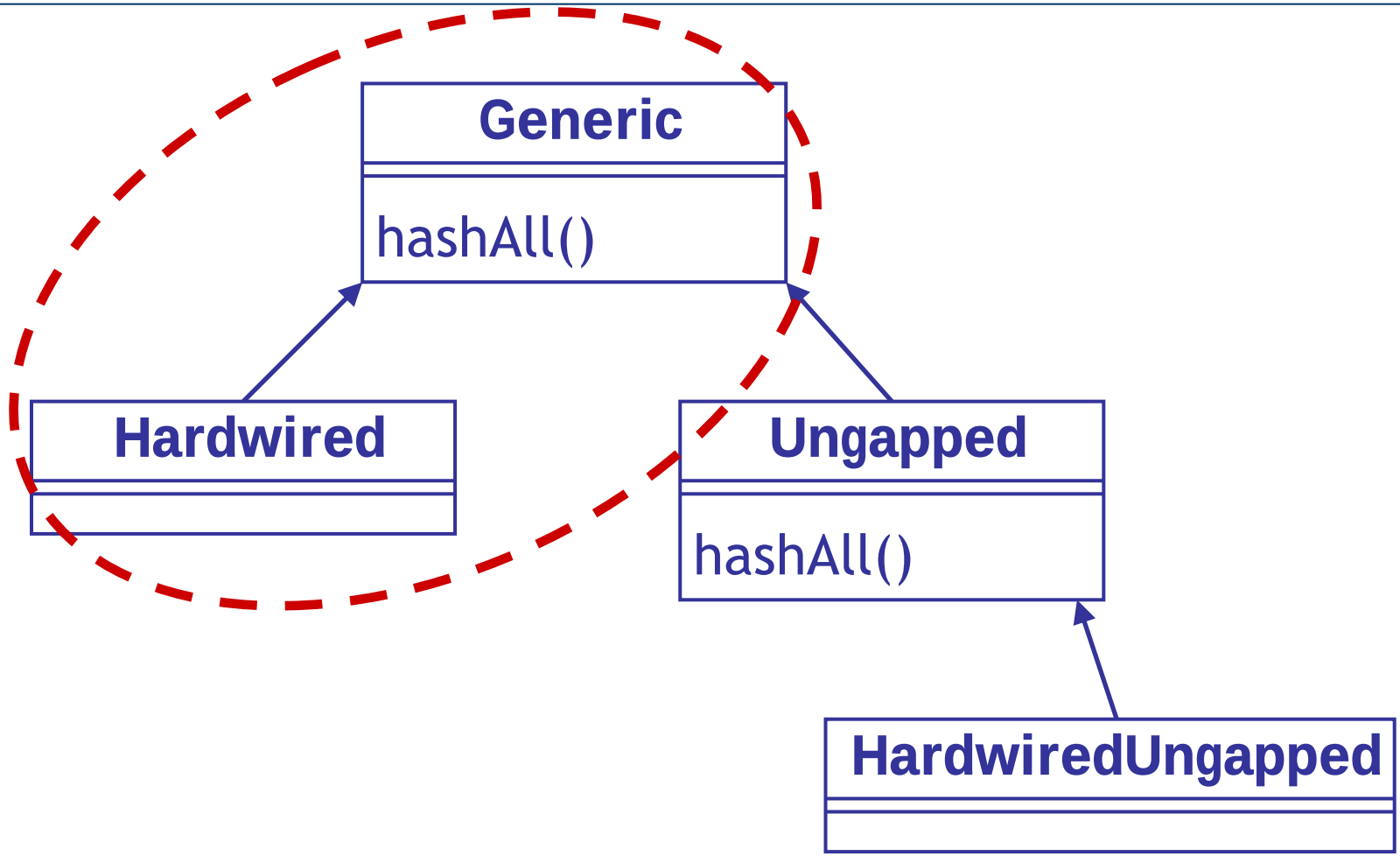
Hardwired

```
hashAll(Sequence, Shape)
{
    Shape.hash(0)
    for position=1 to Sequence.length():
        Shape.hashNext(position)
}
```

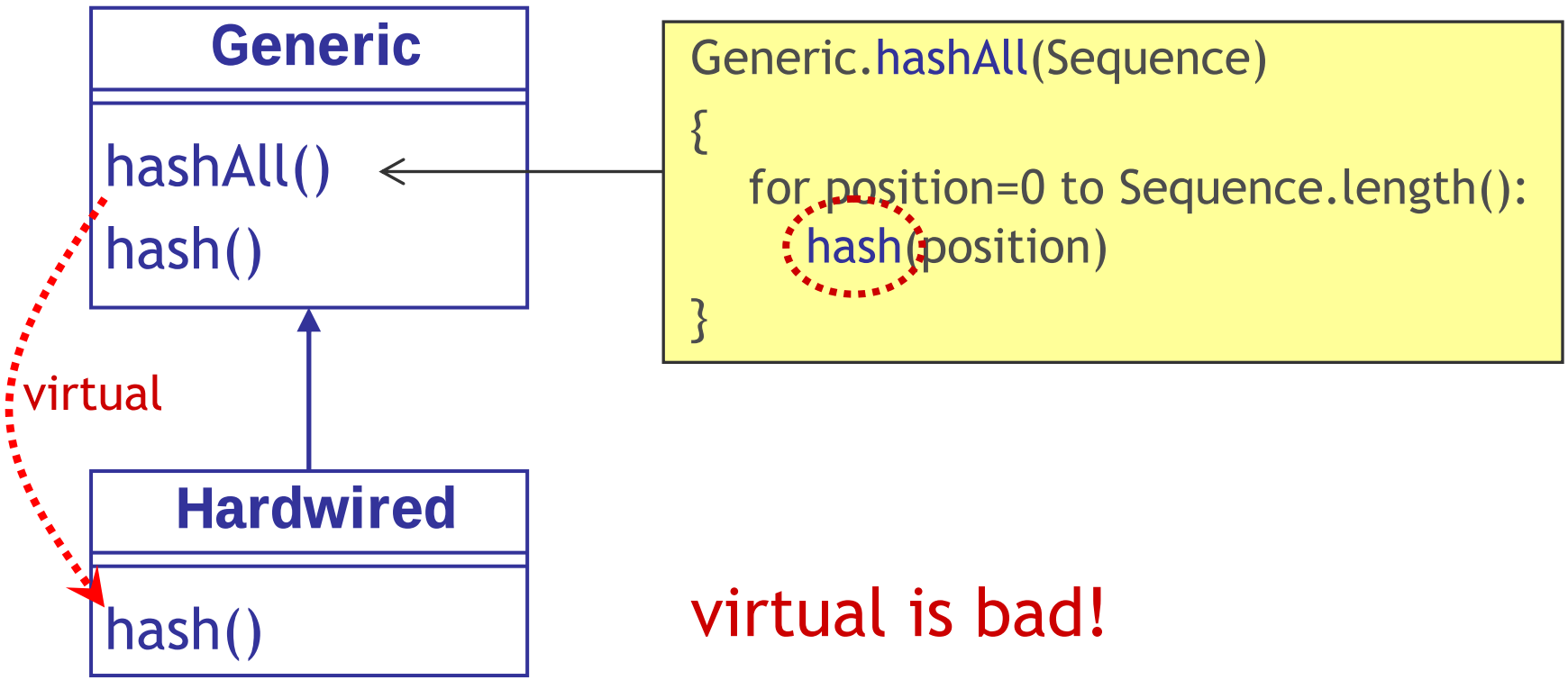
Ungapped

Hardwired Ungapped

OOP


Generic
hashAll()
Hardwired
Ungapped
hashAll()
HardwiredUngapped

Dynamic Binding



virtual is bad!

Wish List

We need:

- Polymorphism
- Delegation (Inheritance)
- Spezialization (Overloading)
- Static Binding

Solution:

C++ Templates

Template Subclassing

Shape ⟨ ⟩

Shape ⟨ Hardwired ⟨ … ⟩ ⟩

Shape ⟨ Ungapped ⟨ ⟩ ⟩

Shape ⟨ Ungapped ⟨ q ⟩ ⟩

Template Subclassing

```
hashAll(seq, Shape ⟨ ? ⟩ sh)
{
    for position=0 to seq.length():
        hash(sh, position)
}
```

Shape ⟨ ⟩

Shape ⟨ Hardwired ⟨ … ⟩ ⟩

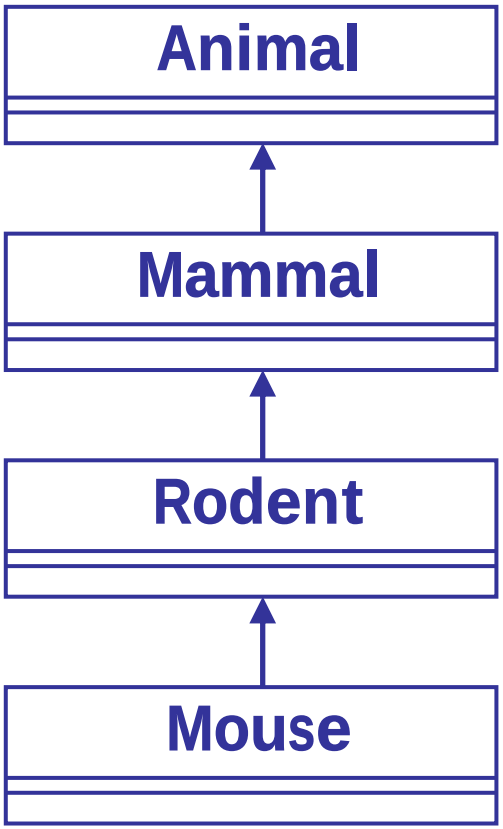
```
hashAll(seq, Shape ⟨ Ungapped ⟨ ? ⟩ ⟩ sh)
{
    hash(sh, 0)
    for position=1 to seq.length():
        hashNext(sh, position)
}
```

Shape ⟨ Ungapped ⟨ ⟩ ⟩

Shape ⟨ Ungapped ⟨ q ⟩ ⟩

Template Subclassing

Subclasses specified by template arguments



Animal ⟨ ⟩

Animal ⟨ Mammal ⟨ ⟩ ⟩

Animal ⟨ Mammal ⟨ Rodent ⟨ ⟩ ⟩ ⟩

Animal ⟨ Mammal ⟨ Rodent ⟨ Mouse ⟩ ⟩ ⟩

Template Subclassing

All animals must eat:

✓ eat(Animal ⟨ ? ⟩)

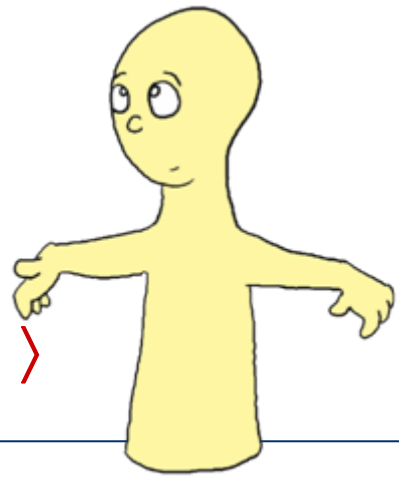
Rodents eat greenstuff:

eat(Animal ⟨ Mammal ⟨ Rodent ⟨ ? ⟩ ⟩ ⟩)

Pandas eat bamboo:

eat(Animal ⟨ Marsupials ⟨ Panda ⟨ ? ⟩ ⟩ ⟩)

Animal ⟨ Mammal ⟨ Human ⟩ ⟩



Template Subclassing

All animals must eat:

eat(Animal ⟨ ? ⟩)

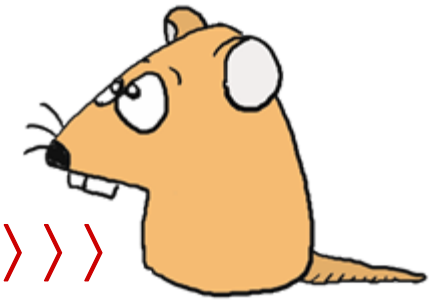
Rodents eat greenstuff:

✓ eat(Animal ⟨ Mammal ⟨ Rodent ⟨ ? ⟩ ⟩ ⟩)

Koala bears eat eucalyptus:

eat(Animal ⟨ Marsupial ⟨ Koala ⟨ ? ⟩ ⟩ ⟩)

Animal ⟨ Mammal ⟨ Rodent ⟨ Mouse ⟩ ⟩ ⟩



Wish List

We need:

- Polymorphism ✓
- Delegation (Inheritance) ✓
- Spezialization (Overloading) ✓
- Static Binding ✓

Conclusion

Message 1:

"Good libraries
offer tailor-made suits of the peg."

Message 2:

"Never put off till run time
what you can do at compile time."

Message 3:

"Use C++ templates
to move work to compile time."

Questions?

EOT

(end of talk)