C++11 CSP
Because we haven’t built enough CSP libraries

Dr Kevin Chalmers

School of Computing
Edinburgh Napier University
Edinburgh
k.chalmers@napier.ac.uk
• Yes you can pull it (although it needs more testing)
• https://github.com/kevin-chalmers/cpp11-csp
1. OK why did you do this?

2. OK, but why did you really do this?

3. But C++ is old hat!

4. How to use C++11 CSP

5. Some Examples

6. Patterns and Networking and Other Stuff

7. Summary
1. OK why did you do this?
2. OK, but why did you really do this?
3. But C++ is old hat!
4. How to use C++11 CSP
5. Some Examples
6. Patterns and Networking and Other Stuff
7. Summary
I started off working with JCSP way back in 2004
  • Undergraduate module in Design Patterns including parallel
  • An iteration of material Jon had been running as occam for years

My undergraduate project looked at code mobility in JCSP
My PhD work looked at networking and networked mobility
One outcome of this work was a generalised protocol for networked channel communications.

I developed a .NET version.
  - Connecting, via channels, JCSP and .NET CSP.

I then set off an occam journey, trying to implement the networking stack.
  - And it was too hard.
  - The networking stack relied too heavily on object-orientation ideas and some data sharing.
Stepping Back - C and MPI

- So I took a step back and thought about redesigning in another language
  - C and C++ being the most likely
  - Cross runtime potential with foreign function interface
  - I’ll get back to this
- As a sidetrack I started exploring MPI to compare against what I had
  - De facto standard for message passing in distributed parallel
- I became interested in MPI as a potential comm layer, but this requires a redesign of networking
  - Too stream focused
- So, I am really trying to create a framework which sits on top of MPI
  - And the simplest method was to move away from Java to C or C++
An aside - C++CSP Already Exists!

- Neil Brown did a lot of work on a C++ version of a CSP library
- Also created C++CSP2
- Also created C++CSP2 Networked
- So why not just use Neil’s work?
- Well, C++CSP2 is not very extensible
  - Couldn’t implement the networked channel model I had easily
  - Would have to strip down C++CSP2 to achieve what I wanted
  - Also relied heavily on Boost and OS specific libraries
1. OK why did you do this?

2. OK, but why did you really do this?

3. But C++ is old hat!

4. How to use C++11 CSP

5. Some Examples

6. Patterns and Networking and Other Stuff

7. Summary
Oh how much do I hate writing JCSP code!
Actually, I hate writing Java code - JCSP does a good job of hiding Javaisms™ (Chalmers 2015)
I want to decide if I am passing by value or reference
I want to decide how and when an object is destroyed
I want to override operators
I could go on...
I like the expressiveness of parallel computation in occam.

- I do - *even though I’ve never been an occam programmer*
- What I don’t like is the lack of libraries
  - As a programmer I find libraries more important than the language
  - I can work round the language with APIs
- I also find occam a little archaic
  - But let’s not start an argument about it
- So I want the simplicity of occam, but with massive library and platform support
- These two issues quite happily collided
1 OK why did you do this?

2 OK, but why did you really do this?

3 But C++ is old hat!

4 How to use C++11 CSP

5 Some Examples

6 Patterns and Networking and Other Stuff

7 Summary
• C++ is an object-oriented language
  • No - but you can write object-oriented code with C++
  • A lot of Java programmers (including me) write bad C++ code
• C++ is too complex
  • Depends on your approach - you can write complex code with C++
  • I like to tell students that C++ is not a forgiving language
• C++ relies on pointers
  • No - most C programmers wrote a lot of bad C++ code
• What C++ is is up for debate
  • It is multi-paradigm. If actor models take off you can guarantee that C++ will get send and receive or similar
New C++ Features

- OK, there are a lot, so I’ll summarise the important bits
  - **Smart pointers** Automatic release of resources - no need for `new` and `delete`
  - **Lambda Expressions** You can do pretty much standard functional language stuff - it just gets pretty ugly
  - **Move semantics** You can now move resources as well as copy and reference
    - Yes, this is just occam-$\pi$ mobile data types
  - **Initializer lists** Create collections using braces to denote the members
  - **Variadic templates** Compiler does more work for the type

- **Concurrency** **FINALLY!** We get
  - Thread
  - Mutex
  - Future
  - etc.
• OK, last slide before getting onto some code
• Modern C++ design encourages the following approaches
  - **Avoid pointers** Most memory should have a handler object that looks after the resource
  - **PIMPL** Private implementation objects surrounded by a handler
  - **RAII** Resource lifetime is linked to the lifetime of its owner
  - **Generic programming** Templates to create reusable code
1. OK why did you do this?
2. OK, but why did you really do this?
3. But C++ is old hat!
4. How to use C++11 CSP
5. Some Examples
6. Patterns and Networking and Other Stuff
7. Summary
Creating and Using Channels

// Channels are typed
one2one_chan<int> c;
// Operator overloads for read and write
auto x = c();
c(x);
// Also overloads to automatically convert to chan_in, chan_out, etc.
Processes are Just Functions

// Lambda expression
auto send = [&]() { c(5); };
auto recv = [&]() { cout << c() << endl; };
// Can also bind functions with values
// void sender(chan_out<int> c)
// void receiver(chan_in<int> c)
fork f1(make_proc(sender, c));
fork f2(make_proc(receiver, c));
f1();
f2();
par uses Initializer Lists

```c
// Parallel now looks like a control block
par
{
    prefix(0, a, b),
    delta(b, {c, d}),
    succ(c, a),
    printer("", ",", d)
}();
```

- I’d like to do something similar with `alt` - have an idea for that
1 OK why did you do this?
2 OK, but why did you really do this?
3 But C++ is old hat!
4 How to use C++11 CSP
5 Some Examples
6 Patterns and Networking and Other Stuff
7 Summary
1. OK why did you do this?
2. OK, but why did you really do this?
3. But C++ is old hat!
4. How to use C++11 CSP
5. Some Examples
6. Patterns and Networking and Other Stuff
7. Summary
Building the pattern functions for C++11 CSP got me interested in parallel patterns
  • I’ll talk about this more tomorrow
I’m almost there with a new networking stack
  • Far more versatile and extensible
  • Will require only little bits of code to enable new comm layers
I’d like automatic fork management (i.e. you fork a process you have to wait for it to complete)
I’d like other syntactic sugar approaches built in (e.g. alt)
Outline

1. OK why did you do this?
2. OK, but why did you really do this?
3. But C++ is old hat!
4. How to use C++11 CSP
5. Some Examples
6. Patterns and Networking and Other Stuff
7. Summary
• This is very much a pet project to let me explore ideas
• I’m not aiming for performance, but simplicity of code
• My main aim is to build a new MPI supported network stack for channel processes
• And hopefully stick that into Go, Rust, etc.
• So, happy enough to share code, but don’t expect support at the moment
Questions?