L26: JCSP (2)

César Sánchez

Grado en Ingeniería Informática
Grado en Matemáticas e Informática
Universidad Politécnica de Madrid

Fri, 22-May-2015

Este texto se distribuye bajo los términos de la Creative Commons License
JCSP in a nutshell

- Channels:

  [Diagram of channels]

  Any2OneChannel
  One2OneChannel

  Any2OneChannel ch = Channel.any2one; // ch : Any2OneChannel
  chOut = ch.out(); // chOut : ChannelOutput
  chIn = ch.in(); // chIn : ChannelInput

- How to send and receive:

  ch.out().write('a');
  chOut.write(count);
  c = (MyClass) chIn.read();

- Recepcion alternativa condicional:

  AltingChannelInput [] the_channels = {ch1.in(), ch2.in()};
  services = new Alternative(the_channels);
  boolean syncCond [] = { true, false };
  switch (services.fairselect(syncCond)) {
    case 0:
      //...
  }
You must answer these questions:
Implementing Resources with JCSP

You must answer these questions:

1. How many channels?
Implementing Resources with JCSP

You must answer these questions:

1. How many channels?

2. How to match resource operations and channels
Implementing Resources with JCSP

You must answer these questions:

1. How many channels?

2. How to match resource operations and channels

3. How to program a CPRE
Implementing Resources with JCSP

You must answer these questions:

1. How many channels?

2. How to match resource operations and channels

3. How to program a CPRE

4. When and how to make a client stop
Implementing Resources with JCSP

You must answer these questions:

1. How many channels?

2. How to match resource operations and channels

3. How to program a CPRE

4. When and how make a client stop

Two properties:

1. Safety: (CPREs are guaranteed when operation takes place)
2. Progress: (some enabled is not blocked)
Encapsulation of invocations

- One *useful* idea is to model resource operations as Java methods...
  
  ... that implement the use of the channels
Encapsulation of invocations

- One *useful* idea is to model resource operations as Java methods...
  
  ... that implement the use of the channels

Example: inc/dec

![Diagram](image)
Encapsulation of invocations

- One *useful* idea is to model resource operations as Java methods…
  … that implement the use of the channels

Example: inc/dec
Encapsulation of invocations

- One *useful* idea is to model resource operations as Java methods...
  ... that implement the use of the channels

Example: inc/dec

option 1: one channel per operation
Encapsulation of invocations

- One *useful* idea is to model resource operations as Java methods... 
  ... that implement the use of the channels

Example: inc/dec

option 2: one method per operation
Encapsulación de invocaciones

- Una idea útil es modelar operaciones de recursos como métodos Java que... que implementan el uso de los canales

Ejemplo: inc/dec

opción 2: un método por operación
Encapsulación de invocaciones

- Una idea útil es modelar operaciones de recursos como métodos de Java...
  ... que implementan el uso de los canales

Ejemplo: inc/dec  

Opción 2: un método por operación
Encapsulation of invocations

- One useful idea is to model resource operations as Java methods...
  ...

Example: inc/dec

option 2: one method per operation

```java
class ContadorJCSP implements CSProcess {
    // ...
    public void incrementar() {
        chIncDec.out().write('i');
    }
}
```
Encapsulation of invocations

- One *useful* idea is to model resource operations as Java methods...
  ... that implement the use of the channels

Example: inc/dec

Option 2: one method per operation

```java
class ContadorJCSP implements CSProcess {
    public void run() {
        while (true) {
            kind = (Character)chIncDec.in().read();
            if (kind == 'i') {
                count++;
            } else { // kind=='d'
                count--;
            }
        }
    }
}```
Useful patterns and idioms

- One server thread that performs the operations sequentially.
- Server reads requests and writes answers. Clients write requests and read answers.
- Sometimes one channel per operation, sometimes encapsulate a sequence of channels to implement an operation.
- For example, to return values:
  - Pass the (one2one) response channel in the request.
- Blocking clients can only be done ignoring their messages!
- But one cannot ignore some messages and not others in a given channel
  ⇒ For different CPREs, use different channels.
- For simultaneous listening to channels, use alternative/fairselect.
How many channels? When to read/write?

Q1. How many channels?

Q2. When and how to read/write from them?
How many channels? When to read/write?

Q1. How many channels?

Q2. When and how to read/write from them?

Answers (art of programming):

A3. It depends on the CPREs.
How many channels? When to read/write?

Q1. How many channels?

Q2. When and how to read/write from them?

Answers (art of programming):

A3. It depends on the CPREs.

   Case 1:   CPRE : True
How many channels? When to read/write?

Q1. How many channels?

Q2. When and how to read/write from them?

Answers (art of programming):

A3. It depends on the CPREs.

   Case 1: CPRE : True

   One global channel
How many channels? When to read/write?

Q1. How many channels?

Q2. When and how to read/write from them?

Answers (art of programming):

A3. It depends on the CPREs.

   Case 1: CPRE : True
       
       One global channel

   Case 2: CPRE ≠ True, and CPRE does not depend on input
How many channels? When to read/write?

Q1. How many channels?

Q2. When and how to read/write from them?

Answers (art of programming):

A3. It depends on the CPREs.

   Case 1: CPRE : True

       One global channel

   Case 2: CPRE \ne True, and CPRE does not depend on input

       One channel per operation, “recepción alternativa condiciona”
How many channels? When to read/write?

Q1. How many channels?

Q2. When and how to read/write from them?

Answers (art of programming):

A3. It depends on the CPREs.

Case 1: CPRE : True

One global channel

Case 2: CPRE ≠ True, and CPRE does not depend on input

One channel per operation, “recepción alternativa condiciona”

Case 3: CPRE ≠ True, and CPRE does depend on input
How many channels? When and how to read/write?

Case 3: \( \text{CPRE} \neq \text{True} \), and CPRE \textbf{does depend} on input
Case 3: CPRE $\neq$ True, and CPRE does depend on input

Sometimes we can reduce to a few cases (by clustering the input values).
Case 3: \( \text{CPRE} \neq \text{True}, \) and CPRE \textbf{does depend} on input

Sometimes we can reduce to a few cases (by clustering the input values).

Two input values are in the \textbf{same equivalence class} precisely when their CPREs are true in the same states of the resource.
How many channels? When and how to read/write?

Case 3: CPRE ≠ True, and CPRE does depend on input

Sometimes we can reduce to a few cases (by clustering the input values).

\[ \text{Two input values are in the same equivalence class precisely when their CPREs are true in the same states of the resource.} \]

Example: multibuffer
How many channels? When and how to read/write?

Case 3: \( \text{CPRE} \neq \text{True}, \) and \( \text{CPRE} \text{ does depend} \) on input

Sometimes we can reduce to a few cases (by clustering the input values).

Two input values are in the \text{the same equivalence class precisely when their CPREs are true in the same states of the resource.}

Example: multibuffer

\[ \left\lfloor \frac{\text{max}}{2} \right\rfloor \text{ equivalence classes} \]

\[ \text{has (potentially) infinite input space} \]
How many channels? When and how to read/write?

Case 3: CPRE $\not= \text{True}$, and CPRE does depend on input

Sometimes we can reduce to a few cases (by clustering the input values).

**equivalence classes**

Two input values are in the same equivalence class precisely when their CPREs are true in the same states of the resource.

Example: multibuffer

- has $\left\lfloor \frac{\text{max}}{2} \right\rfloor$ equivalence classes
- has (potentially) infinite input space

Solution: one CPRE per equivalence class

“Indexing by parameter” ("indexación de parametros")
Case 3: CPRE $\neq$ True, and CPRE does depend on input
How many conditions?

Case 3: \( \text{CPRE} \neq \text{True} \), and CPRE does depend on input

What if the quotient set is huge or infinite? (for example if the CPRE does depend on values)

quotient set: set of equivalence classes
How many conditions?

Case 3: \( \text{CPRE} \neq \text{True} \), and \( \text{CPRE} \) does depend on input

What if the *quotient set* is huge or infinite? (for example if the CPRE does depend on values)

Example: multibuffer with \( 10^9 \) entries.

Example: blocking collection (set) with \( \text{CPRE} : e \in S \)

\[
\text{waitUntil}(e) \\
\text{POST: } self = self^{pre}
\]
How many conditions?

Case 3: CPRE ≠ True, and CPRE does depend on input

What if the quotient set is huge or infinite?
(for example if the CPRE does depend on values)

Example: multibuffer with $10^9$ entries.

Example: blocking collection (set) with
CPRE : $e \in S$
waitUntil(e)
POST: self = self$^{pre}$

Solution:
1. Blocking client creates channel...
2. ...and send channel in request
3. Unblocking traverses the collection...
4. ...and chooses proper channel (client)
How many conditions?

Case 3: \( \text{CPRE} \neq \text{True}, \text{and CPRE does depend on input} \)

What if the \textit{quotient set} is huge or infinite? (for example if the CPRE does depend on values)

Example: multibuffer with \(10^9\) entries.

Example: blocking collection (set) with
\[
\text{CPRE} : e \in S
\]
\[
\text{waitUntil}(e)
\]
\[
\text{POST: } self = self^{pre}
\]

Solution:

1. Blocking client creates channel...
2. ...and send channel in request
3. Unblocking traverses the collection...
4. ...and chooses proper channel (client)

"Indexing by client" ("indexación de clientes")