A Simple File Transfer Protocol

Marie Farrell

Department of Computer Science
Maynooth University

Here, we modularise the Event-B specification of a simple file transfer as described in Chapter 4 of Abrial's book. This protocol ensures the transfer of a file from one site to another. The Figures included below contain modular $\mathcal{EVT}$-specifications corresponding to each model in the Event-B specification.

```
spec  FILE_CTX =
  sort D
  ops n : ℕ1
      f : [ℕ1] → D
end
```

```
spec  FILE_0 =
  FILE_CTX then
  ops b:Bool
      g : [ℕ1] ⇸ D
  . b = TRUE ⇒ g = f
    b = FALSE ⇒ g = ∅
  Events
    Init =
      thenAct g := ∅
              b := FALSE
    final =
      when b = FALSE
      thenAct g := f
              b := TRUE
end
```

Fig. 1. Abstract model of the simple file transfer protocol.

```
spec  FILE_1 =
  FILE_0 then
  ops r : ℕ1
       h : [ℕ1] → D
  . r ∈ [1..n+1]
    h = [1..r-1]◁ f
    b = TRUE ⇒ r = n + 1
  Events
    Init =
      thenAct h := ∅
              r := 1
    final =
      when r = n + 1
    receive =
      when r ≤ n
      thenAct h(r) := f(r)
              r := r + 1
end
```

Fig. 2. The first refinement step.

```
spec  FILE_2 =
  FILE_1 then
  ops s : ℕ1
      d : D
  . s ≤ n + 1
    s ∈ [r..r+1]
    s = r + 1 ⇒ d = f(r)
  Events
    Init =
      thenAct s := 1
              d :∈ D
    receive =
      when s = r + 1
      thenAct h(r) := d
              r := r + 1
    send =
      when s = r
           r ≠ n + 1
     thenAct d := f(s)
              s := s + 1
end
```

Fig. 3. The second refinement step.

```
spec  FILE_CTX1 =
  FILE_CTX then
  ops parity : ℕ → ℤ
  . parity(0) = 0
    ∀ x · x ∈ ℕ ⇒ parity(x + 1) = 1 - parity (x)
    ∀ x, y · y ∈ ℕ ∧ x ∈ [y..y + 1] ∧ parity(x) = parity(y) ⇒ x = y
end
```

```
spec  FILE_3 =
  FILE_CTX1 and FILE_2 then
  ops p : ℤ
      q : ℤ
  . p = parity(s)
    q = parity(r)
  Events
    Init =
      thenAct p := 1
              q := 1
    receive =
      when p ≠ q
      thenAct q := 1 - q
    send =
      when p = q
      thenAct p := 1 - p
end
```

Fig. 4. The third refinement step.

```
spec  FILE_4 =
  FILE_3 then
  ops bd, ba : Bool
      pc, qc : ℤ
  . bd = TRUE ⇒ ba = FALSE
    pc = p
    qc = q
  Events
    Init =
      thenAct bd := FALSE
              ba := TRUE
              pc := 1
              qc := 1
    receive =
      when pc ≠ q
           bd = TRUE
      thenAct qc := 1 - q
              bd := FALSE
              ba := FALSE
    send =
      when p = qc
           ba = TRUE
      thenAct pc := 1 - p
              ba := FALSE
              bd := TRUE
end
```

Fig. 5. The fourth refinement step.