1 New Type Definitions

- BIG_INT : Arbitrary precision integers from zarith library.
- DYADIC : Representation of dyadic rational numbers as pairs of BIG_INT, $d = (x, y)$ corresponds to $\frac{x}{2^y}$

2 Components

The system is composed of three major components:

1. CLIENTPROC – Executes the verified MWU extracted from Coq.
   - **sends** to CLIENTENVINTERFACE:
     (big_int, dyadic) list, pairing strategies with weights
   - **receives** from CLIENTENVINTERFACE:
     (big_int, dyadic) list, pairing strategies with costs in the range [-1, 1]
2. ENVPROC – Evaluates the behavior of an agent's distribution and returns a cost vector associated with each strategy.
   - **sends** to CLIENTENVINTERFACE:
     space-separated string representation of float list, where element i corresponds to the cost incurred by strategy i in the current round.
   - **receives** from CLIENTENVINTERFACE:
     space-separated string representation of float list, where element i corresponds to the weight the agent associates with strategy i in the current round.
3. CLIENTENVINTERFACE – Unverified OCaml program responsible for managing communication between clientProc and envProc.
   - **sends** to CLIENTPROC:
     A (big_int, dyadic) list, pairing strategies with costs in the range [-1, 1], generated by parsing the string received from CLIENTENVINTERFACE.
   - **sends** to ENVPROC:
     A space-separated string representation of a float list, generated from the list received from CLIENTPROC.
   - **receives** from CLIENTPROC:
     A (big_int, dyadic) list, representing the weight vector generated by the client in the current round.
   - **receives** from ENVPROC:
     A space-separated string representation of a list of floats representing the cost vector generated by the environment in the current round.

3 Communication

Communication occurs across two pairs of named pipes, one pair responsible for messages between clientProc and clientEnvInterface, and the other pair handling messages between envProc and clientEnvInterface. Communication to/from clientProc is handled through Ocaml's marshaling functions. Communication to/from envProc simply sends strings.

4 Example of a round

TBD