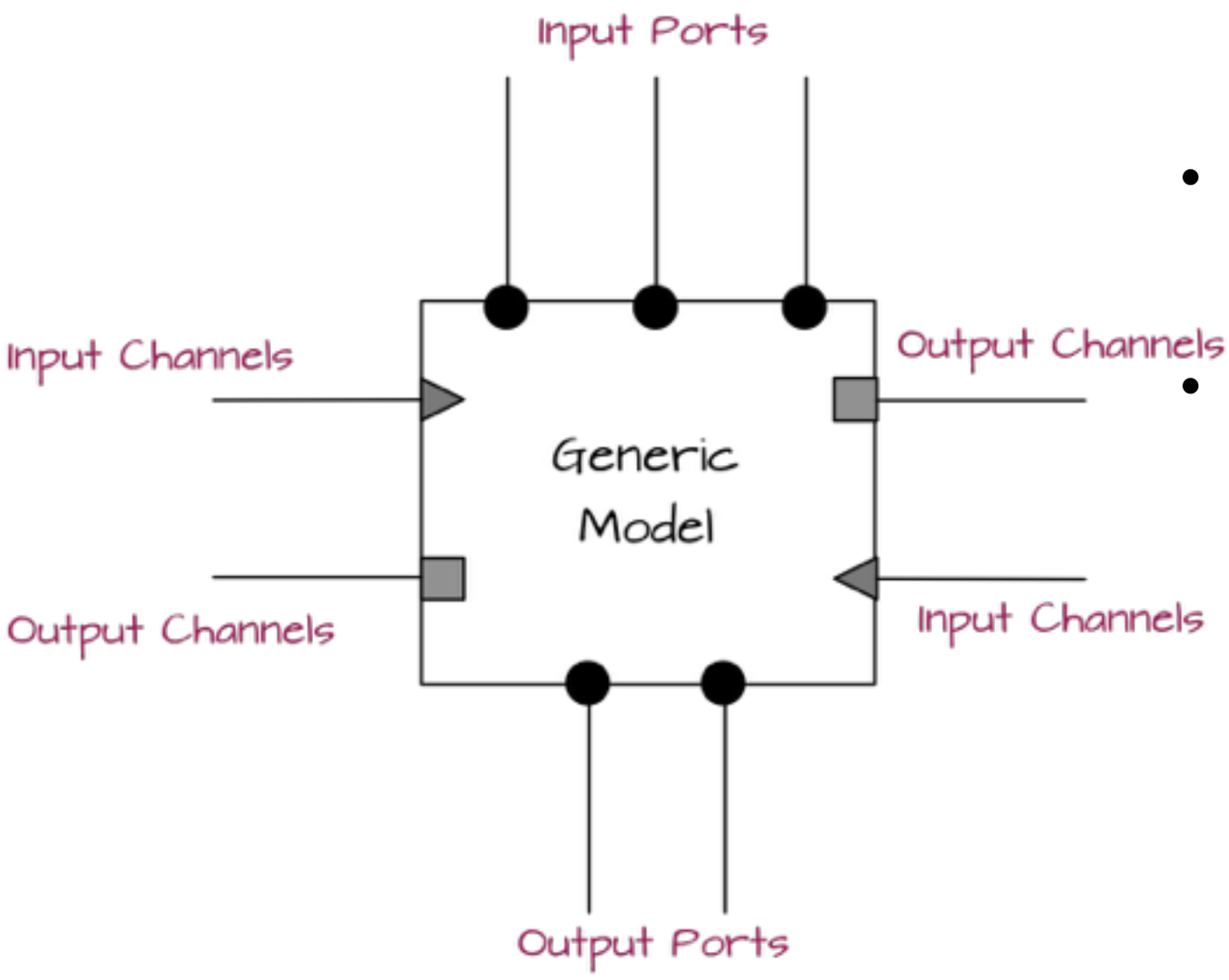
A general hypomodel

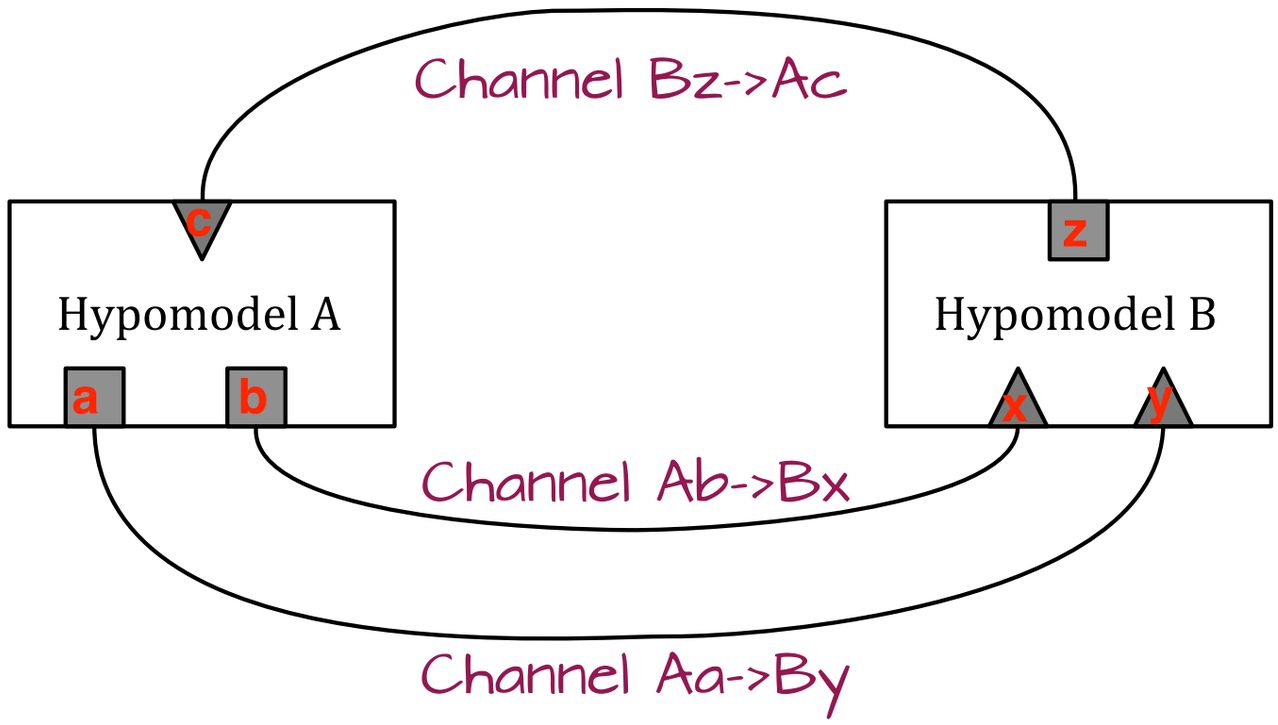


- A set of **inputs**, used for the initialization of its execution
- A set of **outputs** produced and made available after its completion
- A set of communication **channels** with its environments
  - Input channels are used for receiving information from the “environment”
  - Output channels are used for emitting information in the “environment”

Channels

- Each channel is unidirectional i.e. either input or output
- Output channels are *non blocking*
- Input channels are *blocking* i.e. the hypomodel waits until there's a message arrived
- Channels transmit application-specific *messages*

Channels



- Channels are uniquely identified (for a given hyper model) based on the hypomodels they connect, the direction, and the specific channel “ports”.

Example API in C++

https://bitbucket.org/sgsfak/http_msg.git

Example Producer

```
…
chic::Comm com;
com.init(); // initialize the library

// and declare the “global” channel ids, and their local names
com.register_output_channel(“a”, “…unique-id…”);
…
chic::OutChannel ch = com.get_output_channel(“a”);
buffer b = … // message’s payload
ch.put(buffer); // send the message, no waiting/delay
```

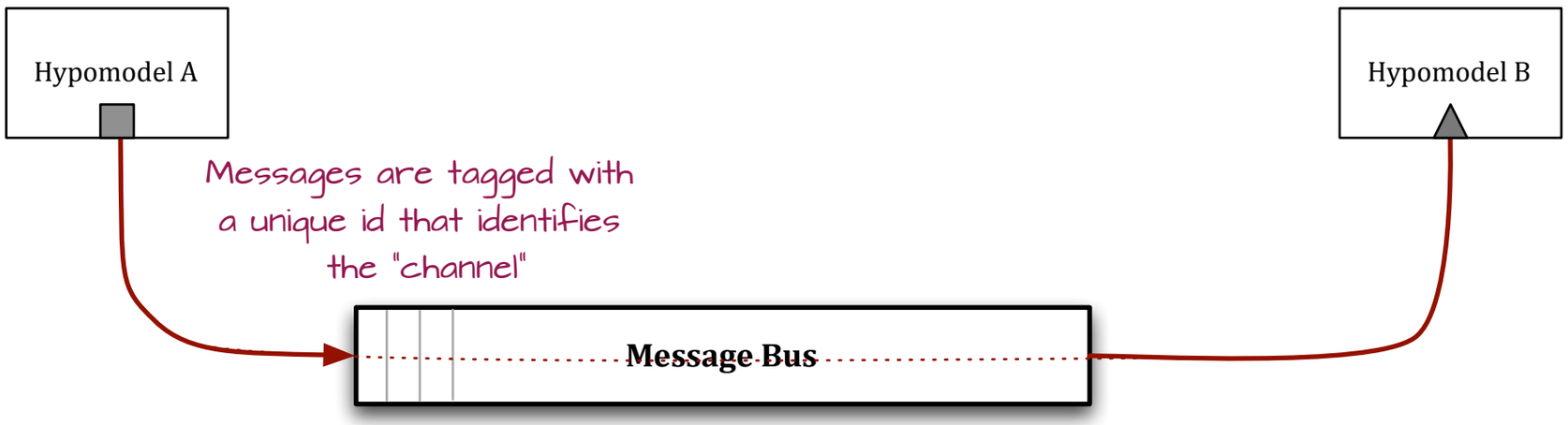
Example Consumer

```
…
chic::Comm com;
com.init(); // initialize the library

// and declare the “global” channel ids, and their local names
com.register_input_channel(“x”, “…unique-id…”);
…
chic::InChannel ch = com.get_input_channel(“x”);
chic::Message msg = ch.get(); // it waits until a message comes

// …alternatively, wait at most some milliseconds, e.g. 1 second:
chic::Message msg;
bool new_msg_recvd = ch.try_get(1000, msg);
if (new_msg_recvd) {
    // …use msg
}
```

Implementation



- Every message is put in a message bus that all hypo models connect to and listen
- The producer and the consumer are decoupled, communication is achieved through the agreement to use the same “channel id”
- A persistent message bus allows for slow consumers to not lose messages

http://qdb.io/