The polarized SILL programming language uniformly integrates functional programming and session-based concurrency. It supports recursion, asynchronous and synchronous communication, and higher-order programs that communicate processes. In this talk, I will motivate and present a denotational semantics for a fragment of polarized SILL. I will explain why domain theory is an ideal setting for this semantics. Session types will be interpreted as domains of bidirectional communications. We use polarity and domain theory to split these into pairs of unidirectional communications in a natural way. Processes will be interpreted as continuous functions on domains of unidirectional communications.