Modular reasoning about concurrent programs is complicated by the possibility of interferences happening between any two instructions of a task (pervasive interference), and these interferences not giving out any information about the behaviors of potentially interfering concurrent tasks (oblivious interference). Reasoning about a concurrent program would be easier if a programmer modularly and statically (1) knows precisely the program points at which interferences may happen (sparse interference), and (2) has some insights into behaviors of potentially interfering tasks at these points (cognizant interference). In this work we present Panini, a core concurrent calculus which guarantees sparse interference, by controlling sharing among concurrent tasks, and cognizant interference, by controlling dynamic name bindings and accessibility of states of tasks. Panini promotes capsule-oriented programming whose concurrently running capsules own their states, communicate by asynchronous invocations of their procedures and dynamically transfer ownership. Panini limits sharing among two capsules to other capsules and futures, limits accessibility of a capsule states to only through its procedures and dispatches a procedure invocation on the static type of its receiver capsule. We formalize Panini, present its semantics and illustrate how its interference model, using behavioral contracts, enables Hoare-style modular reasoning about concurrent programs with interference. We propose three avenues of future work. First, adding capsule inheritance and dynamic dispatch; second, enabling modular reasoning about message orders; and third, enabling modular client reasoning using behavioral contracts.