小步语义

小步语义（Small-step Semantics）是操作语义（Operational Semantics）的一种。可以理解为小步语义定义了一个抽象机器，其描述了在某程序状态下，表达式计算或程序指令执行的**原子**过程。

例如，在某种小步语义下，$(1+2)+(3+4)$ 求值需要三步：先化简左半部分得到 $3+(3+4)$，再化简右半部分得到 $3+7$，最后得到 10。这里所展示的事情是，对于某表达式或者程序指令，总可以根据小步语义定义的规则计算出其运行一步之后的结果；如果找不到就表明计算终止。

与小步语义相对的是大步语义（Big-step Semantics），即指称语义。称它是大步是因为只要给定程序指令和起始状态，那么如果程序终止，就可以找到对应的终止状态，相当于跳过了中间过程、直接知道了结果。

符号表示

由于程序的执行会导致程序状态的改变，因此可以用程序状态和指令的二元组的二元关系表示一步运行过程，记作

$$(c_1, st_1) \to (c_2, st_2)$$

即要执行的程序是 c_1，此时状态是 st_1，那么按照小步语义的规则运行一步，到达的状态是 st_2，剩余待执行程序为 c_2。

方便起见，称小步语义定义的这种二元关系为小步关系。

具体规则

方便起见，不给出具体的规则。只给出一个串行执行规则：

$$(\text{Skip}; c, st) \to (c, st)$$

即串行程序要执行下一条指令，必须等待前一条执行终止。

终止状态

无论是表达式还是程序指令，都应至少有一个终止状态，到达终止状态时表明求值或者程序执行终止。

对于算术表达式，其终止状态是一个数。对于布尔表达式，其终止状态是一个布尔值。

对于程序指令，其终止状态是 Skip。因此某条指令 c 运行一步后执行完了就可以写作

$$(c, st_1) \to (\text{Skip}, st_2)$$

多步关系

小步关系有时不是很有用。我们定义多步关系为其的自反传递闭包。如果 (c_1, st_1) 经多个小步运行到 (c_2, st_2)，则记作

$$(c, st_1) \to^* (c_2, st_2)$$

也即 $((c_1, st_1), (c_2, st_2))$ 在多步关系中。