**Composite内语句块**

composite内部的语句块即为costreamStmts部分，它包括普通c语句、operator定义、composite调用以及splitjion和pipeline等特殊operator定义。costreamStmts部分控制各个operator之间的连接，通过控制连接方式从而构造数据流图。

**operator定义**

operator定义

**composite调用**

COStream支持层次性流图的构造，因此，在composite内部可以调用其他composite。Composite只能进行非递归的调用。具体的调用方式如下：

int a=…

int b=…

stream<int x>P,Q,O1;

stream<float x>O2;

(O1, O2)=COM(P,Q)(a,b);

这里P、Q、O1和O2都是数据流stream变量；a,b为实参。因此composite COM是一个有2个输入流，2个输出流同时带有2个参数的定义。

因此，COStream流程序是由composite文法结构来描述程序代表的数据流图，并通过调用composite来扩展流图结构，实现不同类型的流应用程序，下面的流程序例子反映了这样的语言特点，这里省略了param参数。

*composite M (output K,L, input G,H) {*  //1

stream<int x> I,J,K,L; //2

*I=O(G)()*; //3

*J=P(H)();* //4

*K=Q(I,J)();* //5

*L=R(J)()*; //6

*}* //7

*composite Main{ //8*

*stream<int x>A,B,C,D,E,F*; //9

*…*

*(C,D) = M(A,B)()*; //10

*(E,F) = M(A,B)()* ; //11

*…*

*}*

图1描述了行1至行11的程序例子所反映的流图调用、扩展过程：由Original graph扩展为Expanded graph。Original graph包含了2个composite结构M，第一个M将输入流A和B计算后得到输出流C和D，第二个M将输入流A和B计算后得到输出流E和F。行1定义了M的输入流为G和H，输出流为K和L。行8表示当M在Main中第一次被调用时，输入流和输出流发生实例化：G=A,H=B,K=C以及L=D。在Expand graph中M因为被调用2次而实例化成2份，所以M中的中间流I和J也被实例化为2份：C.I，C.J和E.I，E.J。这些子流图因被调用而实例化展开的过程在编译器编译时完成，即经过编译后，流程序的流图结构将被扩展成完整的流图，程序中的多个composite也将转变为一个。

**

图1： composite流图扩展过程