第 2 卷 第 2,3 期
1997 年 3 月

中国图象图形学报
China Journal of Image and Graphics

Vol. 2, No. 2,3
Mar. 1997

通用可视化平台 GIVE 之数据流核心模型的设计与实现

杨旭波　蔡文立　石教英

(浙江大学 CAD&CG 国家重点实验室，杭州 310027)

摘　要　根据通用科学可视化平台 GIVE 中提出的数据流模型，文中从图论角度出发，运用统一抽象的概念对该模型进行了详细的分析和阐述，然后讨论了其实现技术。

关键词　科学可视化，可视化平台，数据流模型

1　引　言

科学可视化软件平台的研制是科学可视化研究领域内的一个重要分支，是沟通可视化理论与应用的一座桥梁。GIVE(General Interactive Visualization Environment)[1]是我们历时 3 年开发完成的一个通用可视化应用构造工具，GIVE 采用国际上流行的数据流模型为体系结构，并提供交互方便的可视编程界面，用户以“搭积木”的模块级编程方式从模块库中选取所需的可视化模块，直接构造其可视化应用实例，而无须编写代码，有效地提高了软件生产率。GIVE 在数据流模型中引入了控制结点以实现循环和分支功能，能胜任复杂的可视化应用构造。

目前国际上成功的同类平台大都采用数据流模型，如 apE[2]，AVS[3]，IRIS Explorer[4]等。数据流模型的运行方式可分数据驱动和命令驱动两种，apE 和 IRIS Explorer 均采用数据驱动，AVS 采用“lazy evaluation”的命令驱动，它们均只支持一种驱动方式。GIVE 通过对数据流模型的透彻分析，将两种驱动方式统一起来，提供局部数据驱动、局部命令驱动和全局数据驱动 3 种运行操作，使用户对可视化应用的运行控制更加灵活，充分发挥了数据流模型的作用。由于 GIVE 引入了上述 3 个同类平台中所没有的控制结点，因此其数据流模型更具特殊性。

2　GIVE 的数据流模型

2.1　概念和术语

在 GIVE 的数据流模型中，一个应用程序是由多个功能独立的模块通过数据流通道互联构成，每个模块带有若干输入口和输出口，模块的功能是将其输入口的数据进行变换后形成数据流送给与之相连的其它模块。与传统的控制流思想相比，数据流的思想非常适于并行和模块级编程，也非常适于用图示的方式来描述和分析应用程序(图 1)，数据在经过一个个模块的变换后，最后成为几何数据被绘制显示。

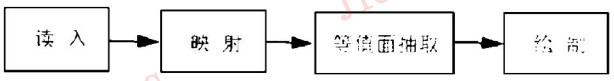

图 1　数据流模型示例

Fig. 1　An example of dataflow model.

结点是相对独立的具有数据变换功能的模块，有入端口和出端口，管道是结点间传递数据的通道。GIVE 中还提供具有分支和循环功能的控制结点，控制结点只控制数据流向，不进行数据变换。流图是

收稿日期：1996 年 10 月；收到修改稿日期：1997 年 1 月。

由结点和管道组成的有向图,一个流图对应一个具体的可视化应用实例,表示各模块间的数据传递和依赖关系。GIVE 的流图可表示为有向图 G≤V,E,F,C,S>:V 是结点集,包括控制结点在内的所有结点;E 是边集,即流图中的管道;F 是映射关系,F:E→V×V;C 是约束条件集,它规定了 G 中各边的合法性,即用户在创建管道时应遵循的规则;S 是状态集,表示每个结点的当前状态。若在图 G 中存在从结点 V1 到结点 V2 的路径,则称 G 中 V1 可达 V2,记为 access(V1,V2)。在 G 中可达 V 的所有结点的集合称为 V 的上游,记为 upstream(V, G)={ V′ | access(V′,V)},其中 V′称为 V 的上游结点。在 G 中 V 可达的所有结点集合称为 V 的下游,记为 downstream(V,G)={V′ |access (V,V′)},其中 V′称为 V 的下游结点。

GIVE 中流图结点的驱动方式有两种:(1)在数据驱动(data driven)方式下,当某结点所有入端口数据到达后,该结点就被自动点火执行,执行完后,其出端口数据由管道传递到所有依赖该结点的入端口(图 2),当源结点 A 点火后,downstream(A,G)中所有结点,即 B,C,D,E 均要执行,在此方式下,常会导致不必要的结点点火;(2)在命令驱动(demand driven)方式下,流图中除了有数据流外,还存在一股与数据流逆向而行的命令流,它源于用户对数据结果的请求。只有当一结点的所有入端口数据均到达,且有命令流到达该结点时,该结点才被点火执行。若用户需要结点 E 的执行结果,则发一请求,命令流由 E 出发,经 B,C 到 A,则 A,B,C,E 4 结点被点火执行,而 D 结点不会被点火,即只有 upstream(E,G)中的结点被点火。因此其执行效率较好。

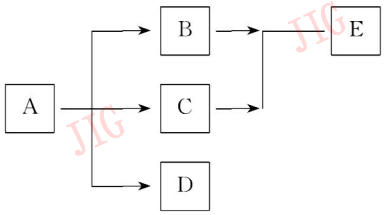

图 2 流图示例

Fig. 2 An example of flow graph.

2.2 流图运行规则

流图中结点有 3 种状态:UPDATED,RUNNING 和 FIRED,其中 UPDATED 表示此结点的运行结果已失效,需重新计算,对于有出端口的结点,意味着其出端口当前数据无效,不能在管道上传送;RUNNING 表示此结点正在运行;FIRED 表示此结点已运行完毕,对于有出端口的结点,意味着其出端口的当前数据有效,可在管道上传送。一个结点是可运行结点的主要条件是:

(1)其状态为 UPDATED;

(2)其所有必选的入端口均有数据;

(3)其所有连有管道的可选入端口均有数据。

可运行结点是否真正运行还要取决于流图运行方式。GIVE 支持 3 种运行操作:

(**1**) 命令驱动方式下的流图局部运行:在此方式下,一个可运行结点真正运行的时机是,用户需要其结果或其下游结点的结果,也就是当用户直接或间接地向运行结点发出请求时,结点才真正运行。故这种方式有时被称为“Lazy Evaluation”方式。在流图 G 中,如 V1 被点火,则在此方式下,请求运行结点集为 upstream(V1,G)∪{V1}。

(**2**) 数据驱动方式下的流图局部运行:在此方式下,用户直接将某一结点 V1 点火,系统运行此结点,并由此结点发出数据流向其下游结点传播,下游结点一旦成为可运行结点就立即运行。其请求运行结点集为 downstream(V1,G)∪ {V1}。

(**3**) 流图的全局运行:在此方式下,流图为“纯数据流驱动”,整个流图从源结点开始,只要是可运行结点就立即运行。其请求运行结点集为{V1|V1∈V}。

用户可随时在这 3 种方式之间任意切换,灵活地控制流图的任意局部运行。事实上,不论何种方式,请求运行结点集中的结点运行都是按数据流驱动原理来控制,集合中任一结点只要是可运行结点,就会立即执行。各结点的运行顺序完全由其何时成为可运行结点所决定。若两结点之间互不可达,即 access(V1,V2)与 access(V2,V1)均不成立,则 V1,V2 可以并行(图 3),当用户在命令驱动方式下,使用结点 F 的点火功能时,请示运行结点集 CRNS={A,B,C,D,F}。在数据驱动方式下,使用结点 B 的点火功能时,CRNS = {B,D,E,F}。当使用全局运行功能时,CRNS ={A,B,C,D,E,F},此时 C 可和{A,B,E}中任一结点并行。

3 数据流模型实现方法

3.1 数据流核心的实现

实现数据流核心时主要考虑 3 方面的问题:

(1)数据流核心采用的体系结构;

(2)界面上的流图结点和实际模块功能如何相结合;

(3)数据流中数据传输的实现机制。

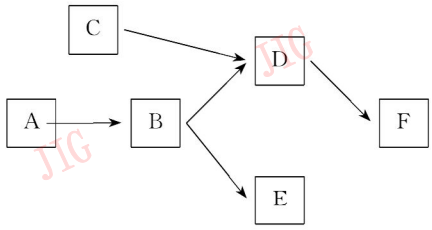

图 3　流图运行规则示例

Fig. 3　An example of running flow graph.

数据流核心体系结构可分为集中式和分散式两种。前者有一总控程序,它保存着流图的全局拓扑信息,以此来控制数据流的运行,后者没有总控程序,每个结点都保存流图的全局拓扑信息,结点之间通过发送和接收消息来相互联系并控制数据流。分散式结构时空耗费较大,且不易控制各结点间的通讯。集中式结构易于实现,时空消耗相对节省,GIVE 采用了集中式结构。

流图结点的功能运行实体可用 4 种方法表示:

(1)进程:流图结点由操作系统进程表示;

(2)函数调用:每个流图结点是一个函数调用接口,结点在运行时动态联编到系统中;

(3)分析器:每个流图结点是送入到一个表达式分析器的字符串,分析器根据给定的语法和语义分析此字符串;

(4)代码:每个流图结点是一组程序语言指令,这组指令在运行时实时地编译或解释执行。GIVE 采用 UNIX 进程,优点在于易于控制,且系统开放性好。当扩充新模块时,只要将新模块封装好并独立编译,就可将其直接用于系统,而系统本身并不要重新编译连接。

GIVE 的总控程序通过 UNIX 的信号机制来控制结点进程,为此定义了 3 种信号:SIGGVRUN 为运行信号,通知结点进程运行;SIGGVSTOP 为暂停信号,通知结点进程暂停。SIGGVTERM 为终止信号,通知结点进程终止。数据流传输采用 UNIX 的共享内存机制来实现,它支持数据在内存中直接传送,数据传送速度快,且能充分利用数据的共享性。

3.2　主要数据结构与算法

GIVE 的数据流核心主要利用模块表、结点表和管道表这 3 个表来记录流图编辑和运行所用信息。

(1)模块表:用于保存系统模块库中每一模块的接口信息(如模块名、执行文件名、端口个数与数据类型、控制参量的说明等),每一表项对应一个模块;

(2)结点表:用于保存流图编辑区中当前所有结点的信息(如结点名、结点的几何位置、结点对应的进程、结点状态、结点所关联的管道等),每一表项对应一个结点;

(3)管道表:用于保存流图编辑区中当前所有管道的信息(如管道所关联的始结点和终结点),每一表项对应一条管道。

实际上,模块表保存了每个模块的所有实例结点的公共信息和静态信息,这些信息最初保存在.spc 文件中,系统启动后将其装入内存的模块表中,在系统运行期间模块表的内容始终不变。结点表中保存了每个结点的动态信息和私有信息,当系统创建某模块的结点实例时,就在结点表中新增一项,并根据模块表的内容填充之。管道表保存了结点之间的数据联系,用户增删管道,系统就修改此表。结点表和管道表共同维护着流图编辑区中当前流图的拓扑结构。有了这 3 个表就能完成流图编辑,流图运行则还依赖于另一重要数据结构——运行表:

```
struct runTerm {
NODEp      NodeP;
    /* 结点指针 */
int        DepNum;
    /* 结点还需要几个结点的运行结果 */
NODEp      DepNodeP[DEPNODES];
    /* 与该结点入端口相连的所有结点 */
Bool       DepFlagA[DEPNODES];
    /* 标记 DepNodeP 表中的结点是否已运行好
*/ }
```

数据流控制核心主要是两个以运行表为核心的算法:调度算法和分派算法。运行表中保存的是请求运行结点集中的结点以及结点的数据依赖关系。调度算法根据 2.2 中的运行规则来组建运行表。分派算法则反复搜索运行表,找出当前可运行结点,将其对应进程激活,结点运行完后,相应地修改运行表内容,找出新的可运行结点,分派算法如下:

```
While (运行表不为空)
    for (运行表中每个可运行结点:即 DepNum=0
        且状态为 UPDATED) do
        if (结点为控制结点) then 计值控制表达
        式,控制数据流向;
        else 向该结点的进程发消息,激活其运行;
```

end for

waitMessage(),即等待从子进程返回的消息;收到消息后,根据消息类别进行处理:

(1)若为 FIRED-MSG,(表示结点运行成功),则 修改运行表中的依赖关系,并从运行表中删去此结点。

(2)若为 ERROR-MSG(表示结点运行报错)则 报错,并退出 while 循环。

(3)若为 EXCEPTION-MSG(表示结点进程出现异常),则 报错,恢复结点进程到正常状态,并退出 while 循环。

End while

其中,可运行结点实现了并行,多个并行进程返回的消息用 UNIX 的消息队列来排队处理。

参考文献

1 杨旭波,蔡文立,石教英. 通用的交互式可视化环境 GIVE. 软件学报,1996,(9):553~558.

2 Dyer,D S, A dataflow toolkit for visualization. IEEE Computer Graphics and Applications, 1990,10(4):60~69.

3 Upson C,Faulhaber T Jr, Kamins D, et al., The application visualization system: A computational environment for scientific visualization. IEEE Computer Graphics and Applications,1989,9(4):30~42.

4 Magaret AnneHalse, IRIS Explorer User's Guide, Document Number 007-137-010,Silicon Graphics Inc.,1992,6.

杨旭波 1971 年生,1992 年本科毕业于国防科技大学计算机系,1995 年硕士毕业于浙江大学计算机系,现为该系博士生。主要研究方向为计算机图形学,科学计算可视化平台。

The Data Flow Model of a General ViSC Platform-GIVE

Yang Xubo Cai Wenli Shi Jiaoying

(*State Key Laboratory of CAD & CG, Zhejiang Univ, Hangzhou* 310027)

Abstract This paper presents the data flow model of GIVE, a general ViSC platform developed by us. From the view of Graph Theory, we discuss this model in detail with unified and abstract concepts. Finally we discuss the implementation of this model.

Keywords ViSC, Platform, Data flow model