LabVIEW: Visual Programming Using a Dataflow Model Extended With Graphical Control Structures.

Brian K. Vogel
Andrew Mihal
{vogel, mihal}@eecs.berkeley.edu
Outline

• Background on LabVIEW
• The "G" language
• LabVIEW demo
Background on LabVIEW

• LabVIEW stands for Laboratory Virtual Instrument Engineering Workbench
• A commercial scientific software system for laboratory automation and simulation
• Developed by National Instruments Corporation
• Users describe a program using a dataflow representation
• LabVIEW uses a visual language called G

• Markets
  – Data Acquisition (DAQ)
  – Test and Measurement (T&M)
  – Industrial Automation (IA)
Background on LabVIEW - User Interface

• Block diagrams
  – Represent the programs
  – Visual dataflow representation
  – Consists of interconnected icons (actors) representing built-in functions, structures, and Virtual Instruments (VIs)
  – Unidirectional edges are called "wires"
• Virtual Instrument (VI)
  – Connected together in a hierarchical block diagram
  – A VI is an actor
  – Can represent a physical laboratory instrument
  – Can interact with physical laboratory instruments via side-effects
Background on LabVIEW - User Interface (cont.)

• Front Panel
  – An execution-time user interface
  – Contains controls (sources) and indicators (sinks)
Overview of the "G" Language

• A visual programming language embedded in LabVIEW

• Based on a dataflow model extended with graphical control flow structures
The G Language - Motivations

• Generality and user convenience (ease of use) are key

• Why extend static dataflow (SDF)?
  – SDF is too restrictive for the typical applications of LabVIEW
  – Control flow constructs would be desirable
The G Language – Motivations (cont.)

• Why not use boolean dataflow?
  – Clarity is a problem
  – Not obvious when or if an arc will carry a data token or a node will execute
  – Cycles, primed arcs, relaxed firing rules are problematic
    • Difficult to understand the computation of a diagram
    • Difficult to construct a correct diagram
  – Control flow is achieved using "goto" statements
Newton’s Method Using BDF

x0

T F Selector

distributor
F T

<

||

-

epsilon

- /

f f’
The G Language – Motivations (cont.)

• G's approach
  – Use structured programming constructs
  – Preserve the clarity of SDF (firing rules, acyclic structure)
  – Enhance clarity by restricting access to code in structure body
The G Language - Overview

• G is a homogeneous, dynamic, multidimensional dataflow language

  • Homogeneous
    – Actors produce and consume a single token from each edge in the graph

  • Dynamic
    – Portions of the graph conditionally executed based on input data

• Multidimensional
  – Support for multidimensional arrays

• Turing Complete
The G Language - Control Structures

• Syntactic representation of a program structure is a box

• Require the box to behave as a node from the outside

• Must satisfy firing rules
  – Data tokens must be available at all input terminals before structure can execute
  – Data tokens are produced for all output terminals when the structure completes

• Structure body behaves as an isolated diagram
The G Language - Control Structures

• Special terminals (called tunnels) are associated with a structure
  – Used to control structure behavior or indicate state
  – Fixed in number
  – Consume/produce data tokens of a particular type
The G Language - Control Structures:

While Loop Structure

- Executes a subdiagram until a continuation flag set by the subdiagram becomes false
- Has two special terminals inside it
  - Count
    - Source of an integer value
    - Initialized to 0 during the first iteration
    - Incremented for each successive iteration
  - Continuation flag
    - Sink for a boolean value
    - A false value causes loop to terminate and output tokens to be sent
While Loop Structure
The G Language - Control Structures:
For Loop Structure

• Executes a subdiagram a predetermined number of times
• Has two special terminals inside it
  – N
    • Specifies number of times to execute the subdiagram
    • An integer sink as viewed from rest of diagram
    • An integer source as viewed from the structure body
  – Count
    • Source of an integer value
    • Initialized to 0 during the first iteration
    • Incremented for each successive iteration
For Loop Structure
The G Language - Control Structures:

Case Structure

- Executes one of several subdiagrams
- Has one special terminal
  - Selector
    - Specifies which subdiagram to execute
    - An integer sink as viewed from rest of diagram
Case Structure
The G Language - Control Structures:

Sequence Structure

- Used to force a certain execution order when no data dependencies exist
- Contains several subdiagrams
- Executes each subdiagram in succession
The G Language - Control Structures:

Shift registers on loop structures

• A program variable that is updated inside the loop

• Consists of a left part and a right part with same vertical location

• Left part is a sink to the outside and a source to loop body

• Data token supplied to right side by subdiagram moved to left side prior to start of next iteration
Shift Registers on Loop Structures

left part

right part

$N$

$i$
LabVIEW demo