Computation graphs

João P. Hespanha

June 13, 2016

1 Computation graph

We are interested in describing a complex computation through a dependency directed bipartite graph. One type of nodes corresponds to multiple-input/multiple-output functions of the form

$$(y_1, y_2, \ldots, y_m) = f(x_1, x_2, \ldots, x_n) \tag{1}$$

and the other type of nodes correspond to variables that are either inputs of outputs to the functions. The node corresponding to the function (1) has input edges from the variables $x_1, x_2, \ldots, x_n$ to f and output edges from f to the variables $y_1, y_2, \ldots, y_m$.

Function nodes without input edges are used to represent constant variables. Variable nodes without input edges correspond to variables that must be computed externally.

The order of the inputs and outputs to the function (1) matters, so the digraphs must have "ordered edges." The variable nodes may have different types, which are defined implicitly by the function they are connected to.

2 File format

The file representation of computational graphs needs to support very large computations, which dictates the need for very efficient and economical representations. Four files are used to represented the computational graphs:

- `filename.cgc` describes the constant variables,
- `filename.cgio` describes the functions used to interact with the computational graph,
- `filename.cg` describes the graph structure,
- `filename.cgs` describes symbolic names for the constants, variables, and functions.

All 4 files files start with a 16-bit "magic" integer that specifies the format in which integers are stored in the file, followed by a sequence of variable-length records. The 8-bit magic number can be one of the following value:

- `16` - indices are stored as 16bit integers with the least significant byte first (little-endian byte ordering)
- `32` - indices are 32bit integers with the least significant byte first (little-endian byte ordering)
- `64` - indices are 64bit integers with the least significant byte first (little-endian byte ordering)

2.1 `.cgc` constants file

Following the "magic" integer, this file consists of a sequence of records, each corresponding to a constant that appears in the computation graph. Each record is of the form:

1. 1 integer specifying the *type* of the constant. All valid types of function should be registered.
2. 1 integer specifying the *length* of the data stored in 8bit bytes.
3. Variable-length list of bytes with the *data* representing the constant (in a format specific to the data type).

2.2 `.cg` graph file

Following the "magic" integer, this file consists of a sequence of records, each corresponding to a function node in the computation graph. Each record is of the following form:

1. 1 integer specifying the *type* of the function. All valid types of function should be registered.

   The special type `0` (zero) is used to denote a constant function that returns a single output equal to one of the functions in the `.cgc` constants file. In this case, the type integer is followed by

   (a) 0-based number of the record where the constant appears in the `.cgc` constants file
   (b) 0-based integer specifying the output variable to the function.

   For this type of function node, the items below do not apply.
2. 1 integer specifying the number of input variables to the function.
3. 1 integer specifying the number of output variables to the function.
4. Variable-length list of 0-based integers specifying the (appropriately ordered) input variables to the function.
5. Variable-length list of 0-based integers specifying the (appropriately ordered) output variables to the function.

2.3 `.cgio` I/O file

Following the "magic" integer, this file consists of a sequence of records, each corresponding to one gateway function used to interact with the computational graph. Each record is of the following form:

1. 1 integer specifying the *type* of the function:

   - `1` - gateway "set" function used set the value of variable nodes.
   - `2` - gateway "get" function used to get the value of variable nodes.
   - `3` - gateway "copy" function used to copy the values of variable nodes to other variable nodes.
2. 1 integer specifying the number of variables to set/get/copy
3. Variable-length list of 0-based integers specifying the (appropriately ordered) variables to set, get, or copy-from (source)
4. Variable-length list of 0-based integers specifying the (appropriately ordered) variables to copy-to (destimation). Only for gateway "copy" functions.

2.4 `.cgs` symbolic name file

Following the "magic" integer, this file consists of a sequence of records, each corresponding to a symbolic name for a constant, variable, or function. Each record is of the following form:

1. 1 integer specifying the type of the record:
    - `1` - constant
    - `2` - function node
    - `3` - variable node
    - `4` - io function
2. 1 integer with the 0-based index of the constant/function/variable
3. 1 integer with number of characters in the symbolic name
4. 1 integer with number of characters in the constant/function/variable description
5. Variable-length string with the symbolic name
6. Variable-length string with the constant/function/variable description

This file is mostly used for debug.