Reach.Coreach Tool: Software Design Description (Draft)

Gordon Marks

June 3, 2020

Table of Contents

Chapter 1

Introduction

This document is a Software Design Description (SDD) for the Reach/Coreach Tool. The purpose of the tool is to perform an impact analysis from a selection of blocks and/or signal lines with the model containing those blocks. The tool is also used to perform model slices to reduce the model to the relevant blocks and signals.

Chapter 2

Top Level Design Overview

The ReachCoreach Tool is primarily focussed in the main ReachCoreach class file contains the bulk of the analyses, but uses a number of functions within to complete a variety of functionalities. The “find” functions (the files named starting with “find” and ending with “RCR”) are used to identify implicit connections between Gotos, Froms, Tag Visibilities, Data Store Reads, Writes, and Memories.

2.1 Detailed Design

2.1.1 sl_customization

```
function sl_customization(cm)
```

The sl_customization file is responsible for handling the options available from the Simulink context menu (when you right-click) under Reach/Coreach.

2.1.2 ReachCoreach

```
classdef ReachCoreach < handle
    % REACHCOREACH A class that enables performing
       reachability/coreachability
    % analysis on blocks in a model.
    %
    %   A reachability analysis (reach) finds all blocks
       and lines that the
```

```
%   given initial blocks affect via control flow or
   data flow. A
%   coreachability analysis (coreach) finds all blocks
    and lines that affect
%   the given initial blocks via control flow or data
   flow. After creating a
%   ReachCoreach object, the reachAll and coreachAll
   methods can be used to
%   perform these analyses respectively, and highlight
    all the
%   blocks/lines in the reach and coreach.
%
% Example:
%       open_system('ReachCoreachDemo_2011')
%       r = ReachCoreach('ReachCoreachDemo_2011');
%
%   % Perform a reachability analysis:
%       r.reachAll({'ReachCoreachDemo_2011/In2'},[]);
%
%   % Clear highlighting:
%       r.clear();
%
%   % Change the highlighting colors
%       r.setColor('blue', 'magenta')
%
%   % Perform a coreachability analysis:
%       r.coreachAll({'ReachCoreachDemo_2011/Out2', {'
   ReachCoreachDemo_2011/Out3'},{});
%
%   % Perform a slice:
%       r.slice();
```

The ReachCoreach file defines the ReachCoreach class, whenever the tool is used, an object of this class will be created. The following nested sections refer to functions defined in the ReachCoreach file.

2.1.2.1 getColor

```
        function [fgcolor, bgcolor] = getColor(object)
```

```
    % GETCOLOR Get the highlight colours for the
       reach/coreach.
    %
    %   Inputs:
    %       object  ReachCoreach object.
    %
    %   Outputs:
    %       fgcolor Foreground colour.
    %       bgcolor Background colour.
    %
    %   Example:
    %       obj.getColor()
```

2.1.2.2 setColor

```
function setColor(object, color1, color2)
    % SETCOLOR Set the highlight colours for the
       reach/coreach.
    %
    %   Inputs:
    %       object  ReachCoreach object.
    %       color1  Parameter for the highlight
       foreground colour.
    %               Accepted values are 'red', '
       green', 'blue', 'cyan',
    %               'magenta', 'yellow', 'black',
       'white'.
    %
    %       color2  Parameter for the highlight
       background colour.
    %               Accepted values are 'red', '
       green', 'blue', 'cyan',
    %               'magenta', 'yellow', 'black',
       'white'.
    %
    %   Outputs:
    %       N/A
    %
    %   Example:
    %       obj.setColor('red', 'blue')
```

2.1.2.3 setHiliteFlag

```
function setHiliteFlag(object, flag)
    % SETHILITEFLAG Set hiliteFlag object property
       . Determines whether
    % to hilite objects or not.
    %
    %   Inputs:
    %       object  ReachCoreach object.
    %       flag    Boolean value to set the flag
       to.
    %
    %   Outputs:
    %       N/A
    %
    %   Example:
    %       obj.setHiliteFlag(true)
```

2.1.2.4 hiliteObjects

```
function hiliteObjects(object)
    % HILITEOBJECTS Highlight the reached/
       coreached blocks and lines.
    %
    %   Inputs:
    %       object  ReachCoreach object.
    %
    %   Outputs:
    %       N/A
    %
    %   Example:
    %       obj.hiliteObjects()
```

2.1.2.5 slice

```
function slice(object)
    % SLICE Isolate the reached/coreached blocks
       by removing
    % unhighlighted blocks.
    %
    %   Inputs:
    %       object  ReachCoreach object.
    %
    %   Outputs:
    %       N/A
    %
    %   Example:
    %       obj.slice()
```

2.1.2.6 clear

```
function clear(object)
    % CLEAR Remove all reach/coreach highlighting.
    %
    %   Inputs:
    %       object  ReachCoreach object.
    %
    %   Outputs:
    %       N/A
    %
    %   Example:
    %       obj.clear()
```

2.1.2.7 reachAll

```
function reachAll(object, selection, selLines)
    % REACHALL Reach from a selection of blocks.
    %
    %   Inputs:
    %       object      ReachCoreach object.
    %       selection   Cell array of strings
       representing the full
    %                   names of blocks.
    %       selLines    Array of line handles.
    %
    %   Outputs:
    %       N/A
    %
    %   Example:
    %       obj.reachAll({'ModelName/In1', '
       ModelName/SubSystem/Out2'}, [])
```

2.1.2.8 coreachAll

```
function coreachAll(object, selection, selLines)
    % COREACHALL Coreach from a selection of
       blocks.
    %
    %   Inputs:
    %       object      ReachCoreach object.
    %       selection   Cell array of strings
       representing the full
    %                   names of blocks.
    %       selLines    Array of line handles.
    %
    %   Outputs:
    %       N/A
    %
    %   Example:
    %       obj.coreachAll({'ModelName/In1', '
       ModelName/SubSystem/Out2'})
```

2.1.2.9 reach

```
function reach(object, port)
    % REACH Find the next ports to call the reach
       from, and add all
    % objects encountered to Reached Objects.
    %
    %   Inputs:
    %       object  ReachCoreach object.
    %       port
    %
    %   Output:
    %       N/A
```

2.1.2.10 coreach

```
function coreach(object, port)
    % COREACH Find the next ports to find the
      coreach from, and add all
    % objects encountered to coreached objects.
    %
    %   Inputs:
    %       object  ReachCoreach object.
    %       port
    %
    %   Outputs:
    %       N/A
```

2.1.2.11 findIterators

```
function iterators = findIterators(object)
    % FINDITERATORS Find all while and for
      iterators that need to be
    % coreached.
    %
    %   Inputs:
    %       object  ReachCoreach object.
    %       port
    %
    %   Outputs:
    %       N/A
```

2.1.2.12 findSpecialPorts

```
function findSpecialPorts(object)
    % FINDSPECIALPORTS Find all actionport,
       foreach, triggerport, and
    % enableport blocks and adds them to the
       coreach, as well as adding
    % their corresponding port in the parent
       subsystem block to the list
    % of ports to traverse.
    %
    %   Input:
    %       object  ReachCoreach object.
    %
    %   Outputs:
    %       N/A
```

2.1.2.13 reachEverythingInSub

```
function reachEverythingInSub(object, system)
    % REACHEVERYTHINGINSUB Add all blocks and
       outports of blocks in the
    % subsystem to the lists of reached objects.
       Also find all interface
    % going outward (outports, gotos, froms) and
       find the next
    % blocks/ports as if being reached by the main
        reach function.
    %
    %   Inputs:
    %       object ReachCoreach object.
    %       system
    %
    %   Outputs:
    %       N/A
```

2.1.2.14 getInterfaceIn

```
function blocks = getInterfaceIn(object, subsystem
   )
    % GETINTERFACEIN Get all the source blocks for
        the subsystem,
    % including Gotos and Data Store Writes.
    %
    %   Inputs:
    %       object      ReachCoreach object.
    %       subsystem
    %
    %   Outputs:
    %       blocks
```

2.1.2.15 getInterfaceOut

```
function blocks = getInterfaceOut(object,
   subsystem)
    % GETINTERFACEOUT Get all the destination
       blocks for the subsystem,
    % including Froms and Data Store Reads.
    %
    %   Inputs:
    %       object      ReachCoreach object.
    %       subsystem
    %
    %   Output:
    %       blocks
```

2.1.2.16 traverseBusForwards

```
function [path, exit] = traverseBusForwards(object
   , creator, oport, signal, path)
    % TRAVERSEBUSFORWARDS Go until a Bus Creator
       is encountered. Then,
    % return the path taken there as well as the
       exiting port.
    %
    %   Inputs:
    %       object  ReachCoreach object.
    %       creator
    %       oport
    %       signal
    %       path
    %
    %   Outputs:
    %       path
    %       exit
```

2.1.2.17 traverseBusBackwards

```
function [path, blockList, exit] =
   traverseBusBackwards(object, iport, signal,
   path, blockList)
    % TRAVERSEBUSBACKWARDS Go until Bus Creator is
        encountered. Then,
    % return the path taken there as well as the
       exiting port.
    %
    %   Inputs:
    %       object      ReachCoreach object.
    %       iport
    %       signal
    %       path
    %       blockList
    %
    %   Outputs:
    %       path
    %       blockList
    %       exit
```

2.1.2.18 addToMappedArray

```
function addToMappedArray(object, property, key,
   handle)
```

```
            % ADDTOMAPPEDARRAY
            %
            %   Inputs:
            %       object      ReachCoreach object.
            %       property
            %       key
            %       handle
            %
            %   Outputs:
            %
```

2.1.3 findFromsInScopeRCR

```
function froms = findFromsInScopeRCR(obj, block, flag)
% FINDFROMSINSCOPE Find all the From blocks associated
   with a Goto block.
%
%       Inputs:
%               obj    The reachcoreach object containing
   goto tag mappings
%       block  The goto block of interest
%       flag   The flag indicating whether shadowing
   visibility tags are in the
%              model
%
%       Outputs:
%               froms    The tag visibility block
   corresponding to input "block"
```

2.1.4 findGotosInScopeRCR

```
function goto = findGotosInScopeRCR(obj, block, flag)
% FINDGOTOSINSCOPE Find the Goto block associated with a
   From block.
%
%       Inputs:
```

```
%               obj    The reachcoreach object containing
   goto tag mappings
%       block  The from block of interest
%       flag   The flag indicating whether shadowing
   visibility tags are in the
%              model
%
%       Outputs:
%               goto   The goto block corresponding to
   input "block"
```

2.1.5 findGotoFromsInScopeRCR

```
function blockList = findGotoFromsInScopeRCR(obj, block,
   flag)
% FINDGOTOFROMSINSCOPE Find all the Goto and From blocks
   associated with a
% Goto Tag Visibility block.
%
%       Inputs:
%               obj        The reachcoreach object
   containing goto tag mappings
%       block      The tag visibility block of interest
%       flag       The flag indicating whether shadowing
   visibility tags
%                  are in the model
%
%       Outputs:
%               blockList  Cell array of goto/from blocks
   corresponding to input "block"
```

2.1.6 findVisibilityTagRCR

```
function visBlock = findVisibilityTagRCR(obj, block, flag)
% FINDVISIBILITYTAG Find the Goto Visibility Tag block
   associated with a
% scoped Goto or From block.
```

```
%
%       Inputs:
%               obj       The reachcoreach object
   containing goto tag mappings
%       block     The goto or from block of interest
%       flag      The flag indicating whether shadowing
   goto tags are in the
%                 model
%
%       Outputs:
%               visBlock  The tag visibility block
   corresponding to input "block"
```

2.1.7 findReadsInScopeRCR

```
function reads = findReadsInScopeRCR(obj, block, flag)
% FINDREADSINSCOPE Find all the Data Store Read blocks
   associated with a Data
% Store Write block.
%
%       Inputs:
%               obj    The reachcoreach object containing
   data store mappings
%       block  The write block of interest
%       flag   The flag indicating whether shadowing data
   stores are in the
%              model
%
%       Outputs:
%               froms    Thedata store read corresponding
   to input "block"
```

2.1.8 findWritesInScopeRCR

```
function writes = findWritesInScopeRCR(obj, block, flag)
% FINDWRITESINSCOPE Find all the Data Store Writes
   associated with a Data
```

```
% Store Read block.
%
%       Inputs:
%               obj     The reachcoreach object containing
   data store mappings
%       block   The read block of interest
%       flag    The flag indicating whether shadowing data
   stores are in the
%               model
%
%       Outputs:
%               froms     The data store write
   corresponding to input "block"
```

2.1.9 findReadWritesInScopeRCR

```
function blockList = findReadWritesInScopeRCR(obj, block,
   flag)
% FINDREADWRITESINSCOPE Find all the Data Store Read and
   Data Store Write
% blocks associated with a Data Store Memory block.
%
%       Inputs:
%               obj         The reachcoreach object
   containing data store mappings
%       block       The data store memory block of interest
%       flag        The flag indicating whether shadowing
   data stores are in the
%                   model
%
%       Outputs:
%               blockList  The cell array of reads and
   writes corresponding to the
%                           input "block"
```

2.1.10 findDataStoreMemoryRCR

```
function mem = findDataStoreMemoryRCR(obj, block, flag)
% FINDDATASTOREMEMORY Find the Data Store Memory block of
   a Data Store
% Read or Write block.
%
%       Inputs:
%               obj    The reachcoreach object containing
   data store mappings
%       block   The data store read or write block of
   interest
%       flag    The flag indicating whether shadowing data
   stores are in the
%              model
%
%       Outputs:
%               mem    The data store memory block
   corresponding to input "block"
```

2.1.11 GroundAndTerminatePorts

```
function GroundAndTerminatePorts(sys)
    % GROUNDANDTERMINATEPORTS Ground and terminate all
       unconnected ports in
    % a system. I.e. For each unconnected input port,
       create a Ground block
    % and connect that block to the port, and for each
       unconnected output
    % port, create a Terminator block and connect that
       block to the port.
    %
    %   Inputs:
    %           sys     Simulink system (fullname or
       handle) for which to
    %                 ground and terminate unconnected ports
       .
    %
    %   Outputs:
    %           N/A
```

2.1.12 hilite_system_notopen

```matlab
function hilite_system_notopen(sys,hilite,varargin)
%HILITE_SYSTEM_NOTOPEN Highlight a Simulink object.
%   HILITE_SYSTEM_NOTOPEN(SYS) highlights a Simulink
   object by WITHOUT opening the system
%   window that contains the object and then highlighting
   the object using the
%   HiliteAncestors property. This is a modification of
   the original function,
%   described below:
%
%   HILITE_SYSTEM_NOTOPEN(SYS) highlights a Simulink
   object by first opening the system
%   window that contains the object and then highlighting
   the object using the
%   HiliteAncestors property.
%
%   You can specify the highlighting options as additional
    right hand side
%   arguments to HILITE_SYSTEM_NOTOPEN.  Options include:
%
%     default     highlight with the 'default' highlight
   scheme
%     none        turn off highlighting
%     find        highlight with the 'find' highlight
   scheme
%     unique      highlight with the 'unique' highlight
   scheme
%     different   highlight with the 'different' highlight
    scheme
%     user1       highlight with the 'user1' highlight
   scheme
%     user2       highlight with the 'user2' highlight
   scheme
%     user3       highlight with the 'user3' highlight
   scheme
```

```
%     user4        highlight with the 'user4' highlight
  scheme
%     user5        highlight with the 'user5' highlight
  scheme
%
%   To alter the colors of a highlighting scheme, use the
  following command:
%
%     set_param(0, 'HiliteAncestorsData', HILITEDATA)
%
%   where HILITEDATA is a MATLAB structure array with the
  following fields:
%
%     HiliteType        one of the highlighting schemes
  listed above
%     ForegroundColor   a color string (listed below)
%     BackgroundColor   a color string (listed below)
%
%   Available colors to set are 'black', 'white', 'red', '
  green', 'blue',
%   'yellow', 'magenta', 'cyan', 'gray', 'orange', '
  lightBlue', and
%   'darkGreen'.
%
%   Examples:
%
%       % highlight the subsystem 'f14/Controller/Stick
  Prefilter'
%       HILITE_SYSTEM_NOTOPEN('f14/Controller/Stick
  Prefilter')
%
%       % highlight the subsystem 'f14/Controller/Stick
  Prefilter'
%       % in the 'error' highlighting scheme.
%       HILITE_SYSTEM_NOTOPEN('f14/Controller/Stick
  Prefilter', 'error')
%
%   See also OPEN_SYSTEM, FIND_SYSTEM, SET_PARAM
```

2.1.13 reachCoreachGUI

```
function varargout = reachCoreachGUI(varargin)
% REACHCOREACHGUI MATLAB code for reachCoreachGUI.fig
%      REACHCOREACHGUI, by itself, creates a new
   REACHCOREACHGUI or raises the existing
%      singleton*.
%
%      H = REACHCOREACHGUI returns the handle to a new
   REACHCOREACHGUI or the handle to
%      the existing singleton*.
%
%      REACHCOREACHGUI('CALLBACK',hObject,eventData,
   handles,...) calls the local
%      function named CALLBACK in REACHCOREACHGUI.M with
   the given input arguments.
%
%      REACHCOREACHGUI('Property','Value',...) creates a
   new REACHCOREACHGUI or raises the
%      existing singleton*.  Starting from the left,
   property value pairs are
%      applied to the GUI before
   reachCoreachGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes
   property application
%      stop.  All inputs are passed to
   reachCoreachGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "
   GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

2.1.14 Diff

The Diff folder in the source of the tool contains the functions related with performing the Reach/Coreach option on models starting from the differences between those models rather than selecting starting points manually.

2.1.14.1 Coreach_Diff

```
function [oldCoreachedObjects, newCoreachedObjects,
   diffTree] = Coreach_Diff(oldModel, newModel, highlight,
    direction, diffTree)
    % COREACH_DIFF Identifies blocks and lines in oldModel
      and newModel that
    % potentially impact the components changed between
     the models.
    %
    % Inputs:
    %   oldModel    The original version of a model.
    %   newModel    The new version of a model.
    %   highlight   [Optional] Indicates whether or not to
      highlight the
    %               differences and impacts. Default: 1 to
      highlight differences
    %               with DarkGreen foreground and Red
     background and highlight
    %               impacts of those differences with
     Yellow foreground and Red
    %               background; use 0 for no highlighting.
    %   direction   [Optional] Indicates direction of
     analysis. Default: 1 for
    %               upstream analysis (Coreach), 0 for
     downstream analysis
    %               (Reach).
    %   diffTree    [Optional] Result of:
    %                   slxmlcomp.compare(oldModel,
     newModel)
    %               Only used to speed up results.
    %
    % Outputs:
    %   oldCoreachedObjects Handles of blocks and lines in
      oldModel that
    %                         potentially impact the changes
       .
    %    newCoreachedObjects Handles of blocks and lines in
        newModel that
    %                         potentially impact the changes
       .
    %    diffTree             Tree generated from:
    %                             slxmlcomp.compare(oldModel
       ,newModel)
    %                         Can be passed back in on
       future calls using the same
    %                         models to speed up results.
    %
```

2.1.14.2 get_diffs_for_reachcoreach

```
function [oldBlocks, oldLines, newBlocks, newLines] =
   get_diffs_for_reachcoreach(model1, model2, diffTree)
    %
    % Inputs:
    %    model1
    %    model2
    %    diffTree     [Optional] Result of:
    %                     slxmlcomp.compare(oldModel,
       newModel)
    %                 Only used to speed up results.
```

2.1.14.3 highlight_model_diffs

```
function [oldBlocks, oldLines, newBlocks, newLines,
   oldSubs, newSubs] = highlight_model_diffs(model1,
   model2, diffTree) %oldBlocks, oldLines, newBlocks,
   newLines)
    %
```

2.1.14.4 model_diff

```
function diffStruct = model_diff(oldModel, newModel,
   diffTree)
    % MODEL_DIFF Performs a diff between 2 models and gets
       changes to blocks,
    % lines, and ports.
    %
    % Inputs:
    %   oldModel    Simulink model.
    %   newModel    Simulink model to treat as an updated
      version of
    %               oldModel.
    %   diffTree    [Optional] Result of:
    %                   slxmlcomp.compare(oldModel,
      newModel)
    %               Only used to speed up results.
    %
    % Outputs:
    %   diffStruct  Struct containing all blocks, lines,
      and ports that have
    %               changed between oldModel and newModel.
       The fields are
    %               explained below.
    %       diffStruct.comparisonRoot - xmlcomp.Edits
      object representing
    %               the model changes using MATLAB's built
      -in structure.
    %       diffStruct.blocks - Struct used to list the
      changed blocks.
    %           diffStruct.blocks.add - Struct used to
      list the added blocks.
    %               diffStruct.blocks.add.new - Cell
      array of blocks added
    %                       to the new model.
    %               diffStruct.blocks.add.old - Cell
      array of blocks added
    %                       to the old model (note
      this should always be
    %                            empty, but is here for
      consistency in the
    %                            diffStruct field structure
       e.g.
    %                            diffStruct.blocks.del.old
      is not always empty).
    %                    diffStruct.blocks.add.all - Cell
      array of blocks added
    %                            to either model.
    %            diffStruct.blocks.del - Struct used to
      list the deleted blocks;
    %                    has the same field structure as
      diffStruct.blocks.add.
    %            diffStruct.blocks.mod - Struct used to
      list the modified blocks;
    %                    has the same field structure as
      diffStruct.blocks.add.
    %            diffStruct.blocks.mod0 - Struct used to
      list the modified blocks
    %                    excluding SubSystems just with
      changed number of ports;
    %                    has the same field structure as
      diffStruct.blocks.add.
    %            diffStruct.blocks.rename - Struct used to
      list the renamed
    %                    blocks;
    %                    has the same field structure as
      diffStruct.blocks.add.
    %        diffStruct.lines - Struct used to list the
      changed lines;
    %                has the same field structure as
      diffStruct.blocks.
    %        diffStruct.ports - Struct used to list the
      changed ports;
    %                has the same field structure as
      diffStruct.blocks.
    %        diffStruct.notes - Struct used to list the
      changed annotations;
```

```
    %               has the same field structure as
       diffStruct.blocks.
    %       Note: blocks are ultimately recorded in a cell
        array, lines and
    %       ports are ultimately recorded in a numeric
       array.
    %
```

2.1.14.5 Reach_Diff

```
function [oldReachedObjects, newReachedObjects, diffTree]
   = Reach_Diff(oldModel, newModel, highlight, direction,
   diffTree)
    % REACH_DIFF Identifies blocks and lines in oldModel
       and newModel that are
    % potentially impacted by the changes made between the
        models.
    %
    % Inputs:
    %   oldModel    The original version of a model.
    %   newModel    The new version of a model.
    %   highlight   [Optional] Indicates whether or not to
        highlight the
    %               differences and impacts. Default: 1 to
        highlight differences
    %               with DarkGreen foreground and Red
       background and highlight
    %               impacts of those differences with
       Yellow foreground and Red
    %               background; use 0 for no highlighting.
    %   direction   [Optional] Indicates direction of
       analysis. Default: 0 for
    %               downstream analysis (Reach), 1 for
       upstream analysis
    %               (Coreach).
    %   diffTree    [Optional] Result of:
    %                   slxmlcomp.compare(oldModel,
       newModel)
    %                   Only used to speed up results.
    %
    % Outputs:
    %   oldReachedObjects   Handles of blocks and lines in
        oldModel that are
    %                       potentially impacted.
    %   newReachedObjects   Handles of blocks and lines in
        newModel that are
    %                       potentially impacted.
    %   diffTree            Tree generated from:
    %                           slxmlcomp.compare(oldModel
       ,newModel)
    %                       Can be passed back in on
       future calls using the same
    %                       models to speed up results.
    %
```

2.1.15 ReachUtility

The ReachUtility folder in the source of the tool is intended for functions that can be used tangentially to the tool, but that do not represent core functionality.

2.1.15.1 highlight_blocks

```
function highlight_blocks(blocks, background, foreground)
```

2.1.16 Utility

The Utility folder is for files from our overall tools utilities folder. If any functions are saved here, they should not be edited unless at least also editing the version in the general tools utility folder.

2.1.16.1 gcbs

```
function sels = gcbs
    %sels = gcbs
    %returns a cell array of all currently selected blocks
    %limited to the subsystem established by GCB.
    %C. Hecker/11Dec06
```

2.1.16.2 gcls

```
function sels = gcls
    % GCLS Get all currently selected lines.
    %
    %   Inputs:
    %       N/A
    %
    %   Outputs:
    %       sels   Numeric array of line handles.
    %
    %   Example:
    %       >> lines = gcls
    %
    %   lines =
    %       26.0001
    %       28.0004
```