MMAlpha Version 2.0 – Reference Manual[1]

Api/Cosi/R2D2/Compsys/Cairn

March 2008

[1]Last update of April 25, 2009

Contents

Chapter 1

Introduction

This document is the reference Manual of MMALPHA. It concerns Version 2.0 of MMALPHA, but is still under construction. It has been (almost) automatically produced from the usage and note statements of the MMALPHA packages. In Chapter 2, we present the main package of MMALPHA. Chapter 3 concerns functions on domains. In Chapter 4, functions related to the static analysis of ALPHA programs are described. etc.

This manual is part of the general documentation of MMALPHA. This documentation can be accessed through the html hierarchy that follows the MMALPHA hierarchy. The origin of this hierarchy is

```
$MMALPHA/doc/index.html
```

Some conventions

Many functions of MMALPHA operate on the Abstract Syntax Tree (AST) of an ALPHA program and return the new transformed AST. For example, evaluating `normalize[sys]` returns a normalized version of the `sys`, which may be any MATHEMATICA expression whose evaluation is the AST of an ALPHA program. The result of this transformation may be stored in another variable, e.g.:

```
x = normalize[sys];
```

and even manipulated by a MATHEMATICA function.

However, MMALPHA provides another, implicit way of manipulating an ALPHA program. When an Alfa program is parsed and loaded using the `load` function (see `?load`), it becomes the *current* ALPHA *program* and is

stored in a variable named $result. By default, functions operate on $result and modify $result, in addition to returning the transformed program. For example,

```
normalize[];
```

puts in $result the normalized version of $result, and returns this value. (This is the reason of the ; ending this command, otherwise, MATHEMATICA would print out the AST on the notebook.) Before modifying $result, this variable is stored in another variable named $program, so that the value of the current AST before the normalization is kept. The `undo[]` command allows $result to be restored to its previous value.

1.1 Content of the Reference Manual

The reference manual is structured in the following way.

1. The mail LaTeX file, `referenceManual.tex`, is located in directory:

   ```
   ../doc/RefefenceManual
   ```

   It contains an introduction (that you are currently reading) and calls to the various documentation packages, which are produced using commands of the `Makedoc` package (see 11.1.1 for details.

2. The reference manual contains the following Chapters.

   - Chapter 2 presents the functions contained in the `Alpha` package.
   - Chapter 3 presents the functions contained in the `Domlib`, `Zpol`, `Visual`, and `Visual3D` packages.
   - Chapter 4 presents the functions contained in the `Static`, and `Semantics` packages.
   - Chapter 5 presents the functions contained in the `SubSystems`, package.
   - Chapter 6 presents the functions contained in the `Matrix`, and `Tables` packages.
   - Chapter 7 presents the functions contained in the `ChangeOfBasis`, `Cut`, `Normalization`, `Reduction`, `Substitution`, and `Substitution` packages. *This chapter has not been proof-read.*

- Chapter 8 presents the functions contained in the `ScheduleTools`, `VertexSchedule`, and `FarkasSchedule` packages. *This chapter has not been proof-read.*
- Chapter 7 presents the functions contained in the `Pipeline`, `Control`, and `PipeControl` packages. *This chapter has not been proof-read.*
- Chapter 9 presents the functions contained in the `Pipeline`, `Control`, and `PipeControl` packages.
- Chapter 10 presents the functions contained in the `ToAlpha0v2`, `Alphard`, `Vhdl2`, and `VhdlTestBench` packages.
- Chapter 11 presents the functions contained in the `MakeDoc`, package.

3. The status of the documentation contained in each package is described in a note at the beginning of the package: when the documentation has been revised, it appears in a note following the subsection of the package. A revision means that I checked that the content of the usage and note statements of the package appear correctly on the reference manual, but does not mean that the documentation itself is correct.

Acknowledgments (Patrice Quinton)

This version of MMALPHA was developed over the years by the contribution of many people. With the risk of forgetting some contributors (sorry for this), here is a list of these contributors:

- Christophe Mauras invented the ALPHA language in his PhD thesis en 1989, and the actual language is almost its initial version.
- The late Hervé Le Verge designed with Christophe the first version of ALPHA (called ALPHA du Centaur, as it was designed using the Centaur system of Inria.) Hervé also designed the first version of the Polylib which is the engine of MMALPHA.
- Doran Wilde designed the first version of MMALPHA, i.e., the initial MATHEMATICA version of our tool. It started in 1989.
- Zbignew Chamski participated to the development of this first version.
- Tanguy Risset was, and still is, the main architect and contributor of MMALPHA.
- Florent Dupont de Dinechin implemented a significant part of MMALPHA (subsystems, semantic analysis, and part of the back end process).
- Patricia Le Moenner implemented the first version of the Vhdl translator.
- Anne-Claire Guillou contributed to the current Vhdl translater, and designed several circuits.
- Sanjay Rajopadhye was a deep inspirator of MMALPHA.
- Fabien Quilleré implemented mainly the C code generator.
- Several indian students contributed to the development of MMALPHA during internships.

I was often asked why MMALPHA was developed using MATHEMATICA. The choice was made in 1989, mainly in order to take advantage of the available formal calculations embedded in this tool. There are disadvantages to this choice, one being that people have to buy this software in order to use MMALPHA. I believe that this is not the main point, however, since MATHEMATICA is available in many universities, and companies interested in our

software could buy it easily. Probably more important is that MATHEMATICA does not fit in the current *culture* of developers.

But from the point of view of a prototype developper, MATHEMATICA is a very nice tool. Our software is available on *all platforms* thanks to the portability of MATHEMATICA[1]. I have never experienced any problem of running out of space, and time constraints have never been a significant issue. Moreover, every 2 years or so, MMALPHA accelerates significantly when I change my laptop[2].

[1] By the way, the part of MMALPHA that creates most portability problems is the Domlib library that is written in C.

[2] The only occasion when this was not true is when I recently changed my Apple MacBook into a MacBook Pro, where I was disappointed to see the performance degrade. But it may be only because I did not recompile Domlib...

Chapter 2

Basic Functions

2.1 Introduction

Basic functions are mainly in the Alpha package.

2.2 The Alpha‘ package

Introduction

The `Alpha.m` package is the main package of Mathematica. It is located in the directory `$MMALPHA/lib/Packages`. It contains the definition of the main symbols and data structures of MMALPHA.

A special word about the `exportAlphaFunctions` function. This function takes as a parameter the autoloadFile. This file is located in

`$MMAlpha/lib/Packages/Alpha/autoload.m`

and is created automatically by the `exportAlphaFunctions`.

At the end of the `Alpha.m`, a few instructions are executed when loading this package. If the `autoloadFile` does not exists, `exportAlphaFunctions` is called and creates it.

The function `exportAlphaFunctions` uses a local variable `contexts` of `Alpha.m` which contains the list of the packages that need to be loaded when using MMALPHA. This variable has to be changed when a new package is added. For each package of `contexts`, `exportAlphaFunctions` creates an entry in the autoload file that declares the corresponding package using the `DeclarePackage` function of Mathematica. In this way, all packages are declared, and when a symbol of the package is evaluated for the first time, the corresponding package is automatically loaded.

The remaining of this chapter describes the functions of the Alpha package.
Documentation revised on April 25, 2008

2.2.1 Alpha

MMALPHA is a Mathematica-based implementation of the ALPHA language. The "Alpha‘" package contains basic commands for the MMALPHA system. The name "Alpha‘" is also a root name for all packages and symbols of MMALPHA.
Defined in file: Alpha.m
Package: Alpha‘

In this package, there is a description of some structures (domain, pol, zpol), of global variables ($coneList, $demoDirectory, $depCone, $library, $masterNotebook, $MMALPHA, $myMasterNotebook, $myNotebooks, $program, $result, $rootDirectory, $schedule, $scheduleLibrary, $testDirectory, $tmpDirectory, $showGraph, $testResult, $testDirectory, $testIdentifier, $testReportFile, $tmpDirectory, $runTime, $vhdlPatternsDir), and finally of some functions. Among these, ashow, ashowLib, asave, asaveLib, getSystem, load, myStart, putSystem, save, saveLib, show, showLib, showMat, start, and systemNames are important for a user. Functions getPart, readAlpha, readDom, readExp, readDom, testFunction, tests, varTypes, writeC, astQ, fileName, openTemporary, getTemporary, packageLink, demoLink, docLink, link, exportAlphaFunctions, and setMMADir are useful for developers. Function writeTex, enTete, and wrap are unused or incomplete.

2.2.2 domain

domain[dim, idx, lpol] is the structure representing a domain. dim is the (integer) dimension of the domain, idx is the list of dimension names and lpol is a list of polyhedra or of Z-polyhedra. See also ?pol or ?zpol.
Defined in file: Alpha.m
Package: Alpha‘

Domains of Alpha are either polyhedra, unions of polyhedra, Z-polyhedra, i.e., intersection of a polyhedron and of a lattice, or unions of Z-polyhedra. Note also that in domains of Z-Polyhedra, the index part is empty.

2.2.3 pol

pol[nbConstraints, nbRays, nbEq, nblines, cons, rays] is the representation of a polyhedron in Alpha. nbConstraints (integer) is the number of con-

straints (equalities and inequalities), nbRays (integer) is the number of rays (including lines), nbEq (integer) is the number of equalities, nblines (integer) is the number of lines, cons (matrix) is the set of equalities and inequalities, and rays (matrix) is the set of rays and lines. Inequalities and rays are represented by lists of integers with a leading 1, and equalities or lines by lists of integers with a leading 0. See also ?domain and ?zpol.
Defined in file: Alpha.m
Package: Alpha‘

2.2.4 zpol

zpol[m, lpol] is the representation of a Z-polyhedron in Alpha. m is an Alpha matrix which represents the Z-lattice, and lpol is a list of polyhedra. See also ?domain and ?pol.
Defined in file: Alpha.m
Package: Alpha‘

2.2.5 $coneList

$coneList contains the list of dependence cones of all the dependences of $result. This variable is set by the initUniformization function, and it is a list of domains.
Defined in file: Alpha.m
Package: Alpha‘

2.2.6 $demoDirectory

$demoDirectory contains the path of the demo directory of MMALPHA.
Defined in file: Alpha.m
Package: Alpha‘

2.2.7 $depCone

$depCone is a glocal variable which contains the dependence cone of $result. This variable is set by the initUniformization function, and is in domain.
Defined in file: Alpha.m
Package: Alpha‘

2.2.8 $library

$library is the Mathematica symbol which holds the list of Alpha programs (systems and subsystems) most recently loaded using load. The system in $result is also present in $library.
Defined in file: Alpha.m
Package: Alpha‘

2.2.9 $masterNotebook

$masterNotebook is the path of the master.nb notebook, which gives access to all demo notebooks on Alpha. To open it, use the command NotebookOpen[$masterNotebook] or start[].
Defined in file: Alpha.m
Package: Alpha‘

2.2.10 $MMALPHA

$MMALPHA contains the path of the MMALPHA directory. This variable is set by an environment variable called MMALPHA.
Defined in file: Alpha.m
Package: Alpha‘

2.2.11 $myMasterNotebook

$myMasterNotebook is the name of the notebook that you have placed in the $myNotebooks directory to access your own examples. To open it, evaluate myStart[].
Defined in file: Alpha.m
Package: Alpha‘

2.2.12 $myNotebooks

$myNotebooks contains the path of the directory containing your examples. You can set this path in your init.m file. This file, placed in your home directory, is loaded by Mathematica before any other Mathematica command.
Defined in file: Alpha.m
Package: Alpha‘

2.2.13 $program

$program is a Mathematica symbol which keeps the abstract syntax tree of the source of any transformation. It is initialized by load, and changed when a (second) transformation is applied. The most recent AST is kept in $result (see ?$result)
Defined in file: Alpha.m
Package: Alpha‘

2.2.14 $result

$result is the Mathematica symbol which holds the result of the most recent operation (load or transformation). It is the default source program for many transformations. When it is used by default by a transformation, it is also modified by this transformation
Defined in file: Alpha.m
Package: Alpha‘

2.2.15 $rootDirectory

$rootDirectory contains the path of the MMALPHA directory. It is set to $MMALPHA
Defined in file: Alpha.m
Package: Alpha‘

2.2.16 $schedule

$schedule is the Mathematica symbol that contains the schedule information of a system after the execution of the schedule or the scd functions. The format of $schedule is:

scheduleResult[name,
List[{var, varIndices, sched[tauVect, constCoef]}],
objFunction].

where name is the name of the system, var is a variable of sys, varIndices is the list of indexes of var, tauVect is a list of integer corresponding to the linear part of the schedule of var, constCoef is the affine part, and objFunction is the objective function that was used to find the schedule.
Defined in file: Alpha.m
Package: Alpha‘

2.2.17 $scheduleLibrary

$scheduleLibrary is the list of the schedules that have been computed for various programs since the beginning of the current session.
Defined in file: Alpha.m
Package: Alpha‘

2.2.18 $showGraph

$showGraph is a global variable that is set by the report function. Use is not garanteed.
Defined in file: Alpha.m
Package: Alpha‘

2.2.19 $testResult

$testResult is a global variable that is used to store the accumulated result value of a test.
Defined in file: Alpha.m
Package: Alpha‘

2.2.20 $testDirectory

$testDirectory is the path name of the directory that contains the test files for all packages. Currently set to $MMALPHA/tests/
Defined in file: Alpha.m
Package: Alpha‘

2.2.21 $testIdentifier

$testIdentifier contains the name of the test that one wants to perform. Default value is ””.
Defined in file: Alpha.m
Package: Alpha‘

2.2.22 $testReportFile

$testReportFile is the name of the reporting file for the tests. Its value is ”” when no report file is open, otherwise, it contains the
Defined in file: Alpha.m
Package: Alpha‘

2.2.23 $tmpDirectory

$tmpDirectory is the pathname of the directory that will be used for all temporary files, i.e. /tmp/ on unix, C:/tmp/ on WindowsNT (strongly recommended).
Defined in file: Alpha.m
Package: Alpha‘

2.2.24 $runTime

$runTime is a global variable that stores the running time of the scheduler.
Defined in file: Alpha.m
Package: Alpha‘

2.2.25 ashow

ashow[sys] pretty prints in array notation the program contained in sys. Default value of sys is $result. See also show.
Defined in file: Alpha.m
Package: Alpha‘

2.2.26 ashowLib

ashowLib[lib] prints in array notation all the systems of a library lib. Default value of lib is $library. Warning, currently does not work on notebooks, use ashow[lib] instead. See also show, ashow.
Defined in file: Alpha.m
Package: Alpha‘

2.2.27 asave

asave[sys,filename] saves the array notation pretty printed version of $result in file filename. Default value of sys is $result. See also save, load, saveLib, asaveLib.
Defined in file: Alpha.m
Package: Alpha‘

2.2.28 asaveLib

asavelib[lib,filename] saves in array notation all systems contained in library lib in file filename. Default value of lib is $library. See also save, asave.

Defined in file: Alpha.m
Package: Alpha‘

2.2.29 fshow

obsolete form of show.
Defined in file: Alpha.m
Package: Alpha‘

2.2.30 getPart

getPart[exp,position] returns the subexpression of exp which is designated by position. The parameter position is a list of integers following the convention of the Mathematica function Position. For example, getPart[exp,{2,3}] identifies the third subtree of the second subtree of exp. Warning: getPart does not accept in second parameter a list of positions such as those returned by the Mathematica function Position.
Defined in file: Alpha.m
Package: Alpha‘

2.2.31 getSystem

getSystem[sys,lib] extracts the system named sys from the library lib and selects it as the current system. Default value of lib is $library. See also putSystem.
Defined in file: Alpha.m
Package: Alpha‘

2.2.32 load

load[filename] parses the ALPHA program contained in file filename and returns the corresponding abstract syntax tree. As a side effect, symbols $program and $result are loaded with the parsed program. See also save, asave, asaveLib.
Defined in file: Alpha.m
Package: Alpha‘

2.2.33 myStart

myStart[] opens the $myMasterNotebook notebook placed in the directory $MMALPHA/myNotebooks.

Defined in file: Alpha.m
Package: Alpha‘

2.2.34 on

on[] is an obsolete form of start[]. It opens the Master Notebook.
Defined in file: Alpha.m
Package: Alpha‘

2.2.35 putSystem

putSystem[sys,lib] puts the system sys (default $result) into library lib (default $library). If sys already exists in lib, it is replaced, otherwise it is appended. See also getSystem.
Defined in file: Alpha.m
Package: Alpha‘

2.2.36 readAlpha

readAlpha[filename] parses the ALPHA program in file filename and returns its abstract syntax tree. Does not modify $program nor $result.
Defined in file: Alpha.m
Package: Alpha‘

2.2.37 readDom

readDom[dom] parses the ALPHA domain dom and returns its abstact syntax tree. To parse a parametric domain such as {i|i<N}, it is mandatory to supply the parameter domain or the parameter names as a second argument, for example:

readDom[”{i,j | ... }”, {”N”}]
readDom[”{i,j | ... }”,domain[1,{”N”}, ...]] .

Defined in file: Alpha.m
Package: Alpha‘

2.2.38 readExp

readExp[exp] parses the ALPHA expression exp and returns its abstact syntax tree. readExp[exp,paramNames] or readExp[exp,paramDom] are used to parse a parametric expression. Example:

readExp[”A.(i,j,N→N,N) + B.(i,N→i,N)”]

readExp[”A.(i,j→N) + B.(i→i)”, {”N”}]
readExp[”A.(i,j→N) + B.(i→i)”, domain[1,{”N”}, ...]].
Defined in file: Alpha.m
Package: Alpha‘

2.2.39 readMat

readMat[dep] parses the ALPHA dependency dep and returns its matrix abstract syntax tree. To parse a parametric dependency such as (i,j→N), it is mandatory to supply the parameter domain or the parameter names, e.g.:
readMat[”(i,j,N→i)”]
readMat[”(i,j→i)”, {”N”}]
readMat[”(i,j→i)”,domain[1,{”N”}, ...]] .
Defined in file: Alpha.m
Package: Alpha‘

2.2.40 save

save[sys,filename] saves the standard notation pretty printed version of sys in file filename. Default value of sys is $result. See also load, asave, saveLib, asaveLib.
Defined in file: Alpha.m
Package: Alpha‘

2.2.41 saveLib

savelib[lib,filename] saves all systems contained in library lib in file filename in standard notation.
Defined in file: Alpha.m
Package: Alpha‘

2.2.42 show

show[var] pretty-prints the program, the domain, the matrix or the schedule contained in symbol var. Default value of var is $result. show[var, p] pretty prints the domain or the matrix contained in symbol var taking parameter domain p into account. var and p should be abstract syntax trees.
Defined in file: Alpha.m
Package: Alpha‘

2.2.43 showLib

showLib[lib] prints all the systems of library lib in standard notation. Default value of lib is $library. Warning, currently does not work in notebooks, use show[lib] instead
Defined in file: Alpha.m
Package: Alpha‘

2.2.44 showMat

showMat[m] pretty-prints the matrix m in Alpha format.
Defined in file: Alpha.m
Package: Alpha‘

2.2.45 start

start[] opens the Master Notebook.
Defined in file: Alpha.m
Package: Alpha‘

2.2.46 systemNames

systemNames[] returns the list of system names loaded in $library.
Defined in file: Alpha.m
Package: Alpha‘

2.2.47 testFunction

testFunction[expr,resOfExpr,message_String] is used for building tests. It evaluates expr and compare it the resOfExpr, return True if the two are equal, False otherwise and prints a standard message (using the information of message: usually a test number to identify the test). This function is for developers.
Defined in file: Alpha.m
Package: Alpha‘

2.2.48 tests

tests[PackageName] call the test procedure for the package PackageName. tests[] calls the test procedure for all MMALPHA packages. Warning: a complete test takes a long time.

Defined in file: Alpha.m
Package: Alpha‘

2.2.49 varTypes

varTypes[sys] lists the types of all variables of sys. Default value of sys is $result.
Defined in file: Alpha.m
Package: Alpha‘

2.2.50 writeC

writeC[sys,f,opts] generates C code from the system sys in file f. Default value of sys is $result and default value of f is ”Alpha.c”. opts are options that are sent to the C generator. Option ”-p num1 num2 ...” sets Alpha parameters to value num1, num2, etc. The C code is correct only if all Alpha parameters are assigned values. The ”-g” option provides a debug version, where all equation calls are printed out. The ”-s” option is for the generation of C code to be interfaced with the Signal language. WARNING: writeC does not work properly for systems with unbounded domains.
Defined in file: Alpha.m
Package: Alpha‘

2.2.51 writeTex

writeTex[sys,f] generates the Latex form of the program contained in symbol sys (default $result) into file f (default ”Alpha.tex”) of the current directory. writeTex[...,”-a”] produces a program in array notation. Warning: writeTex overwrites an already existing output f file.
Defined in file: Alpha.m
Package: Alpha‘

This function is available only on Unix, and seldom used. Its output is not guaranteed.

2.2.52 astQ

astQ[exp] is True if expression exp is an AST, False otherwise.
Defined in file: Alpha.m
Package: Alpha‘

2.2.53 fileName

fileName[{__String}] returns the path expression corresponding to the file according to the operating system in use. Actually, this function is not necessary, since Mathematica converts all pathnames from the Unix form.
Defined in file: Alpha.m
Package: Alpha‘

2.2.54 openTemporary

openTemporary[] works as the Mathematica function OpenTemporary, but opens the file in the $tmpDirectory and return the corresponding stream. The command Close[openTemporary[] returns the name of the temporary file (warning: there is an inconsistency in this function due to Mathematica between the Unix version and the WindowsNT version: on Unix, the full name returned is ("/tmp/...") while on WindowsNT, only the name of the file is returned. Please use getTemporaryName[] to get the name of a temporary file instead.
Defined in file: Alpha.m
Package: Alpha‘

2.2.55 getTemporaryName

getTemporaryFile[] returns the full name of a new temporary file. It is equivalent to Close[OpenTemporary[]] on a unix plateform and it has the same behaviour on other plateforms.
Defined in file: Alpha.m
Package: Alpha‘

2.2.56 demoLink

demoLink[demo] creates an hyperlink button aiming at the Alpha demo notebook "demo.nb". This button may then be cut and pasted anywhere. Example:

demoLink["Fir"]

creates a button aiming at the demo notebook Fir.nb placed in directory $MMALPHA/demos/NOTEBOOKS/Fir.

demoLink[d,demo] creates an hyperlink button aiming at notebook demo.nb in directory d of $MMALPHA/demos/NOTEBOOKS.
Defined in file: Alpha.m
Package: Alpha‘

2.2.57 docLink

docLink[file] creates an hyperlink button aiming at the Alpha documentation notebook "file.nb". This button may then be cut-and-pasted anywhere. docLink[d:_String, file:_String] creates an hyperlink button aiming at notebook file.nb in directory d of $MMALPHA/doc/packages.
Defined in file: Alpha.m
Package: Alpha‘

2.2.58 link

link[file] creates an hyperlink button aiming at the Alpha notebook "file.nb" in directory $myNotebooks. This button may then be cut-and-pasted anywhere. link[d, file] creates an hyperlink button aiming at notebook "file.nb" in directory d of $myNotebooks.
Defined in file: Alpha.m
Package: Alpha‘

2.2.59 exportAlphaFunctions

exportAlphaFunctions[] writes DeclarePackage commands for all functions of the MMALPHA standard packages. This function is automatically executed when loading Alpha.m (see after the EndPackage instruction) whenever the autoload.m does not exists, and creates this file. The autoload.m file is located in the Alpha directory. Each time you want to add a new package to MMALPHA, add an entry in the local variable "contexts" of Alpha.m, then remove the autoload.m file: a new one will be created.
Defined in file: Alpha.m
Package: Alpha‘

2.2.60 setMMADir

setMMADir[List[___String]] does the SetDirectory[] function work, but the path is specified as a list of strings and hence is valid on different plateforms (Windows, Unix).
Defined in file: Alpha.m
Package: Alpha‘

Actually this function is not needed as Mathematica changes on the fly the path names depending on the platform.

2.2.61 wrap

wrap[exp,contextFile,result] evaluates the expression exp in the context given in the file contextFile, and returns the result in the file result. This function is not currently finished.
Defined in file: Alpha.m
Package: Alpha‘

wrap is used for running examples in separate sessions of Mathematica, for example, using a remote MMALPHA installation. The idea would be to control the execution, and to use only one licence to run these examples. The price to pay would be the start time of Mathematica.

2.2.62 enTete

enTete[symb1_] prints out the standard skeleton for programming a new function named symb1 in MMALPHA (for developper use only).
Defined in file: Alpha.m
Package: Alpha‘

2.2.63 mute

mute is an option (Boolean). If True, function prints absolutely no information.
Defined in file: Alpha.m
Package: Alpha‘

2.2.64 test1

test1[] executes part 1 of tests. See documentation in doc/Tests.
Defined in file: Alpha.m
Package: Alpha‘

2.2.65 test2

test2[] executes part 2 of tests. See documentation in doc/Tests.
Defined in file: Alpha.m
Package: Alpha‘

2.2.66 test3

test3[] executes part 3 of tests. See documentation in doc/Tests.
Defined in file: Alpha.m
Package: Alpha‘

2.2.67 test4

test4[] executes part 4 of tests. See documentation in doc/Texts.
Defined in file: Alpha.m
Package: Alpha‘

Chapter 3

Functions on domains

3.1 The Alpha‘Domlib‘ package

Documentation revised on August 10, 2004

3.1.1 DomLib

Domlib is a library of domain functions. It contains the functions const2al, const2mma, dom2mma, dom2al, domCompRays, DomAddRays, DomBasis, DomConstraints, DomConvex, DomCost, DomDifference, DomEmpty, DomEmptyQ, DomEqualities, DomEqualQ, DomExtend, DomImage, DomIntersection, DomIntEmptyQ, DomLeftHermite, DomLTQ, DomMatrixSimplify, DomPreimage, DomProject, DomRays, DomRightHermite, DomSimplify, DomSort, DomUnion, DomUniverse, DomUniverseQ, DomVertices, DomVisual, DomZImage, DomZPreimage, hypercube, LatticeDifference, LatticeImage, LatticeIntersection, LatticeHermite, LatticePreimage, linearConstraintQ, linearExpQ, linHalfSpace, polToZpol, rays, vertices, and zpolToPol.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

Domlib is based upon the PolyLib library, which is a set of public domain C programs first developed at Irisa. Domlib is interfaced to this library using MathLink.

3.1.2 domlib

Math link. Link to the external program "domlib". The mechanism of MathLink is pretty complicated and needs to be clarified here.

Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.3 const2al

const2al[ind,c] translates in Alpha form the constraint c. ind is an index list (e.g. {"i","j"}) and c is a constraint in form eq[i+2j,3] or ge[i+2j,3].
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.4 const2mma

const2mma[indexList,vectorList] converts a constraint, expressed as a list of index names and a list of vectors, into Mathematica form
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.5 dom2mma

dom2mma[d] converts Alpha domain d into a pair {constraints,index} (e.g. {{i,j},{i+j>=2,...}}) suitable for Mathematica.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.6 dom2al

dom2al[{cst,ind}] translates a domain given in MMA form into its Alpha encoding. ind is a list of indexes, and cst is a list of constraints of the form i+j $>$ 2. Warning: left-hand side is the linear part, and right-hand side is an integer.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

In dom2al, the symbols must be strings, otherwise there is a problem. This has to be checked.

3.1.7 domCompRays

domCompRays[dom] recomputes the rays of the Alpha domain dom and returns an Alpha domain. domCompRays ignores the ray part of dom, and recomputes it. This function allows one to modify a contraint in a domain,

and to update the domain accordingly.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.8 domHalfSpaceQ

domHalfSpaceQ[dom] is True if the Alpha domain dom is a half-space. dom is either a string or an ast.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.9 DomAddRays

DomAddRays[dom,m] returns the domain dom augmented with rays from the matrix m. rays are in Alpha format (i.e. n+2 components for n dimensions).

Example:
dom is {i,j | i >= 0},
m=matrix[3, 4, {}, {{1, 0, 1, 0}, {1, 10, 10, 1}}],
DomAddRays[dom,m] returns {i,j | 0<=i<=(j,10)}.

Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.10 DomBasis

DomBasis[m] returns a row basis of the Alpha matrix m using Gauss Jordan-elimination.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.11 DomConstraints

DomConstraints[m] returns the minimum convex polyehdron defined by the constraint matrix m. DomConstraints[m1,m2] returns the Z-polyhedron obtained by image of the polyhedrom P by the invertible mapping m1, where P is the minimum convex Polyhedron satisfying the constraints given in m2, i.e., Z = m1(DomConstraints[m2]).
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.12 DomConvex

DomConvex[d] returns the minimum convex polyhedron which encloses the (polyhedral or Z) domain d. Warning (4/11/98), this function was bugged in Polylib, and the current implementation uses DomRays instead of DomConvex.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.13 DomCost

DomCost[d, c] returns the interval of values of the cost function c evaluated over domain d = {MinN, MinD, MinI, MaxN, MaxD, MaxI}.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

DomCost is a function of the Domlib library, and what it does in not clear.

3.1.14 DomDifference

DomDifference[d1, d2] returns the domain difference of domain d1 less d2.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.15 DomEmpty

DomEmpty[n] returns the empty domain of dimension n.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.16 DomEmptyQ

DomEmptyQ[d] returns True if domain d is empty, False otherwise. The test of emptyness is based uniquely on the rational polyhedron, not the integral polyhedron.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.17 DomEqualities

DomEqualities[d] returns the matrix of equations (lineality space) of domain d (does not handle union of convexes).

Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.18 DomEqualQ

DomEqualQ[d] returns True if domain d1 is equivalant to d2, False otherwise.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.19 DomExtend

DomExtend[dom, idx] extends dom to the indices in the index list idx. This list should contains the indices of dom.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.20 DomImage

DomImage[d,m] returns the image of the domain d under the transformation matrix m. This function always returns a polyhedral domain. If d is a Z-Domain, it returns the image of the rational polyhedral domain enclosing the Z-Domain.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.21 DomIntersection

DomIntersection[d1, d2] returns the domain intersection of domains d1 and d2.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.22 DomIntEmptyQ

DomIntEmptyQ[dom1, dom2] returns True if dom1 contains no integral points in the context of dom2, False otherwise. ??
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.23 DomLeftHermite

DomLeftHermite[m] returns {H, Q} where m = HQ, Q unimodular, H hermite. (Warning, bugged function please use hermiteL[]).
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.24 DomLTQ

DomLTQ[dom1, dom2, idx, pdim] compares dom1 and dom2 at index position idx (integer). pdim is the dimension of the parameter space. Returns 1 if dom1>dom2, returns -1 if dom1<dom2, returns 0 if dom1><dom2 (whatever it means.)
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

DomLTQ does not check the type and number of parameters.

3.1.25 DomMatrixSimplify

DomMatrixSimplify[m1,m2] returns m1 simplified in the presence of m2.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.26 DomPreimage

DomPreimage[d,m] returns the preimage of the domain d under the transformation matrix m. This function always returns a polyhedral domain. For a ZDomain, it returns the pre-image of the rational polyhedral domain enclosing the ZDomain.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.27 DomProject

DomProject[dom, idx] projects dom onto the indices given in the index list idx. Reorders indices and changes the dimension accordingly.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.28 DomRays

DomRays[m] returns the minimum convex polyhedron defined by the rays given in the matrix m.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.29 DomRightHermite

This function is bugged, use hermiteR instead. DomRightHermite[m] returns the Hermite decomposition {Q, H} of the Alpha matrix m, i.e. m = QH, Q unimodular, H Hermite.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.30 DomSimplify

DomSimplify[d1, d2] returns a domain equal to d1 simplified in the context of d2. In other words, we remove of the definition of d1 all constraints which are implied by d2.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.31 DomSort

DomSort[ldom, idx, pdim, time:True|False] returns the topological ordered list of domains ldom. idx is the level to consider for sorting, pdim is the parameter space dimension. Returns a list of logical times (one per domain) if time is True, otherwise it returns a permutation of ldom. An application of this permutation to ldom returns a sorted list.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

This function seems to be fragile. The only place where it is used is in the INorm package.

3.1.32 DomUnion

DomUnion[d1, d2] returns the domain union of domains d1 and d2.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.33 DomUniverse

DomUniverse[n] returns the universe domain of dimension n.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.34 DomUniverseQ

DomUniverseQ[d] returns True if domain d is the universe, False otherwise.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.35 DomVertices

DomVertices[ldom, context] finds the parametrized vertices of a list of parametrized polyhedra ldom. context is the domain of parameters. It returns a list of pairs whose elements contain a parameter domain, and a list of parametrized vertices. DomVertices[pol, param] finds the parametrized vertices of a parametrized polyhedron pol. DomVertices[pol], finds the vertices of the non-parametrized polyhedron pol. It returns {{e,l}} where e is the universe 0-domain and l is a list of non-parametrized vertices.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

This function seems to be fragile. It is used nowhere in MMALPHA.

3.1.36 DomVisual

DomVisual[d1,d2] visualizes the domain d1 with the Opera tool. d2 is the parameter domain. Warning if the domain d2 is ommited, it is replaced by the parameter domain of $result. WARNING: CURRENTLY NOT AVAILABLE (the function does nothing).
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.37 DomZImage

DomZImage[dom, mat] finds the Z-image of the domain dom by the matrix mat. This function may or may not return a Z-Domain depending on the transformation matrix.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.38 DomZPreimage

DomZPreimage[dom, mat] finds the Z-Preimage of the domain dom by the matrix mat. This function may or may not return a Z-Domain depending on the transformation matrix.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.39 hypercube

hypercube[n] creates a hypercubic domain (in Mathematica form) of dimension n, and size 10. dom2l[hypercube[n]] allows such a domain to be translated into Alpha form. Useful for testing purposes.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.40 LatticeDifference

LatticeDifference[m1,m2] returns the difference of the lattices m1 and m2.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

Seems to return a list of matrices.

3.1.41 LatticeImage

LatticeImage[m1,m2] returns the image of the lattice m1 by the function m2.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.42 LatticeIntersection

LatticeIntersection[m1,m2] returns the intersection of the lattices m1 and m2.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.43 LatticeHermite

LatticeHermite[m] returns the lattice m in Hermite Normal Form.
Defined in file: Alpha/Domlib.m

Package: Alpha‘Domlib‘

3.1.44 LatticePreimage

LatticePreimage[m1,m2] returns the preimage of the lattice m1 by the function m2.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.45 linearConstraintQ

linearConstraintQ[c,{symbols}] is True if c is a linear constraint formed with symbols, False otherwise.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.46 linearExpQ

linearExpQ[exp,{symbols}] is True if exp is a linear expression formed with symbols, False otherwise.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.47 linHalfSpace

linHalfSpace[h] returns the Mathematica Matrix corresponding to the Alpha half-space h.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.48 polToZpol

If the domain d is a union of polyhedra, polToZpol[d] returns the equivalent Z-polyhedron, otherwise it returns d as is.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.49 rays

rays[d] returns the list of rays of the Alpha domain d. rays[{const,index}] returns the vertices of d given in Mathematica form.

Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.50 vertices

vertices[d] returns the list of vertices of the Alpha domain d. vertices[{const,index}] returns the vertices of d given in Mathematica form.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.51 zpolToPol

If the domain d is a Z-Domain, zpolToPol[d] returns the rational polyhedral domain enclosing d, otherwise it returns d as is.
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.52 zpolIsPolQ

zpolIsPolQ[d] is True if the Z-polyhedron d is actually a polyhedron (i.e. has identity matrices as lattices).
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.53 DomTrueRays

DomTrueRays[dom] returns the list of true rays of dom, i.e. rays which are not lines
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.54 DomConstraintsOfDom

DomConstrainsOfDom[dom] returns the constraints
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.1.55 DomLines

DomLines[dom] returns the lines of dom
Defined in file: Alpha/Domlib.m
Package: Alpha‘Domlib‘

3.2 The Alpha‘Zpol‘ package

Documentation revised on August 10, 2004

3.2.1 Zpol

The Alpha‘Zpol‘ package contains additional functions for computing with Z-polyhedra. These functions are expressLatticeWithMat, expressZpolWithMat, getMatOfZpol, and getPolOfZpol.
Defined in file: Alpha/Zpol.m
Package: Alpha‘Zpol‘

The internal representation of a Zpol is: domain[dim_Integer, {___String}, {___zpol[m_matrix,{___pol}]}] (which is m(p)).

3.2.2 expressLatticeWithMat

expressLatticeWithMat[sys,m] returns a system where all Z-polyhedra using lattices corresponding to the matrix m are rewritten so as to appear with matrix m as lattice. Default value of sys is $result. expressLatticeWithMat[sys,m,pos] modifies only the Z-domain at position pos of the Alpha abstract syntax tree.
Defined in file: Alpha/Zpol.m
Package: Alpha‘Zpol‘

3.2.3 expressZpolWithMat

expressZpolWithMat[z,M] attempts to change the representation of the Z-polyhedron z. Assuming z = N(P), we expect a result of the form w = M(P') where M is the matrix given as second argument, hence it returns M(P') where P'=M$\hat{}${-1} N(P). If M and N do not represent the same lattice, it gives a warning and return z unchanged. If z is a union of Z-polyhedra, it fails.
Defined in file: Alpha/Zpol.m
Package: Alpha‘Zpol‘

3.2.4 getMatOfZpol

getMatOfZpol[z] returns the matrix used for identifying the lattice of a Z-polyhedron z. If z=M(P), it returns M. If Z is already a polyhedron, it

returns the identity matrix. If z is a union of Z-polyhedra, it fails.
Defined in file: Alpha/Zpol.m
Package: Alpha‘Zpol‘

3.2.5 getPolOfZpol

getPolOfZpol[z] returns the polyhedron used as source for generating a Z-polyhedron z. If z=M(P), it returns P. If z is already a polyhedron, it returns z. If z is a union of Z-polyhedra, it fails.
Defined in file: Alpha/Zpol.m
Package: Alpha‘Zpol‘

3.3 The Alpha‘Visual‘ package

Documentation revised on August 10, 2004

3.3.1 Visual

The Alpha‘Visual‘ packages contains a few functions to visualize 1D or 2D domains. The main function is showDomain. Function getBoundingBox is also used in some other packages, as well as function bbDomain, which is poorly written.
Defined in file: Alpha/Visual.m
Package: Alpha‘Visual‘

3.3.2 bbDomain

bbDomain[d] returns a list representing the bounding box of domain d. This has the form {{xmin,xmax},{ymin,ymax},{xneg,xpos}, {yneg,ypos}} where {xmin,xmax},{ymin,ymax} is the bounding box of the vertices, {xneg,xpos}, {yneg,ypos} indicate that the domain is infinite in one of these directions. If xneg and xpos are 0, no rays. If xneg is 1, negative ray, if yneg is 1, positive ray. Same for yneg and ypos
Defined in file: Alpha/Visual.m
Package: Alpha‘Visual‘

3.3.3 getBoundingBox

getBoundingBox[dom] gives the bounds of the smallest rectangle containing dom. This function is temporary, in particular it does not handle domains

that are union of polyhedra (in fact it work only on convex polyhedra. It should be reimplemented using DomProject).
Defined in file: Alpha/Visual.m
Package: Alpha‘Visual‘

3.3.4 showDomain

showDomain[d] displays a plot of any 1D or 2D domain or any list of 1D or 2D domains. showDomain[d, name] displays d with the title name specified.
Defined in file: Alpha/Visual.m
Package: Alpha‘Visual‘

3.4 The Alpha‘Visual3D‘ package

Documentation revised on August 10, 2004

3.4.1 Visual3D

The Alpha‘Visual3D‘ package contains a few functions to visualize and animate 3D domains. The main function is vshow, other functions are facets, and listPlanes. Some functions are exported only for debugging purposes.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.2 facets

facets[dom] computes the list of polygons corresponding to the domain dom.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.3 listPlanes

listPlanes[l,p] returns the pair {l,lp} where lp is the lists of planes in p which contain point l.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.4 maxR

maxR is an option of vshow. It allows the viewpoint of the final picture to be set. See stepR for more details.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.5 minR

minR is an option of vshow. It allows the r parameter of the viewpoint to be changed. By default, minR is 1. See also stepR and maxR, and ViewPoint of Mathematica.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.6 orderPolygon

only here for debugging purposes.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.7 stepR

stepR is an option of vshow. It allows the step of the r parameter of the viewpoint to be changed, when one wants the domains to be animated. By default, stepR is 1. Combined with minR and maxR, it allows a sequence of pictures with viewpoint between minR and maxR, by steps stepR to be drawn.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.8 threeDDomainQ

only here for debugging purposes.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.9 twoDDomainQ

only here for debugging purposes.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.10 vp1

vp1 is an option of vshow, allowing the first parameter of ViewPoint to be changed. Default value is 3.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.11 vp3

vp3 is an option of vshow, allowing the third parameter of ViewPoint to be changed. Default value is 1.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.12 vshow

vshow[d] shows a 2D or 3D graphic picture of the 3D domain d. vshow[var] shows the domain of variable var in $result (see Options[vshow] for more details).
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

3.4.13 units

only here for debugging purposes.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

Chapter 4

Static Analysis

4.1 The Alpha‘Static‘ package

Documentation revised on August 10, 2004

4.1.1 Static

The Alpha‘Static‘ package contains functions for the static analysis of Alpha programs. These functions are: analyze, dep, and checkUseful.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

4.1.2 analyze

analyze[sys] performs a static analysis of system sys (default $result) and returns True if the analysis is successful (even with warnings), False otherwise. The analyze function has options verbose, recurse, library and scalarTypeCheck (see Options[analyze]). Option verbose (Boolean; default True) reports analysis progress. Option recurse (Boolean; defaut False) recursively looks up for all the subsystems involved and analyzes them, too. Option library (List; default $library) gives the library to search for subsystems. Example of option usage: analyze[recurse→True, verbose→False]
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

4.1.3 dep

dep[sys] generates a dependency table for sys (default $result). The form of the table is:
dtable[{
depend[Domain, LHS_var_name, RHS_var_name, Matrix, RHS_var_domain],
depend[Domain, LHS_var_name, RHS_var_name, Matrix, RHS_var_domain],
. . .
].
The table may be pretty–printed using show[dep[]].
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

4.1.4 dep1

dep1[sys, {occur:{_Integer..}, occurs___}] generates dependency table entries for all occurrences in the list. Used by dep[] to generate the dependency table.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

4.1.5 checkUseful

checkUseful[sys, var] checks that the variable var is used in the system sys for all the points of its domain. Default value of sys is $result. If var is omitted, the function is applied to all variables of sys.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

4.1.6 checkDeclarations

This function is a private function of the package.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

4.1.7 buildLHSIdList

This function is a private function of the package.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

4.1.8 buildRHSIdList

This function is a private function of the package.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

4.2 The Alpha‘Semantics‘ package

Documentation revised on August 10, 2004

4.2.1 Semantics

Alpha‘Semantics‘ is the package which contains the functions for computing the type of Alpha expressions. These functions are changeType, expDimension, expDomain, expType, getContextDomain, matchTypes, and replaceByEquivExpr.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

4.2.2 changeType

changeType[sys,var,newType] changes the type of the variable var to newType in sys (default $result). No typing compatibility check is done: this is not a semantic preserving transformation. Mainly used for changing integers to fixed bit width integers.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

4.2.3 expDimension

expDimension[sys,exp] computes the dimension of the expression exp the in system contained in the system sys (default $result). The expression may be given as a string, a position vector (following the convention of the Mathematica function Position) in the system, or as an AST. expDimension outputs error messages if the dimensions are not compatible, and returns dimension -1 in this case. Remark: parameters are counted as additional dimensions.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

4.2.4 expDomain

expDomain[sys,exp] computes and returns the domain of exp in program sys (default $result) or $Failed if the expression is badly formed. expDomain generates error and warning messages accordingly. The expression may be given as a string, a position vector (following the convention of the Mathematica function Position) in the system, or as an AST.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

4.2.5 expType

expType[sys,exp] computes the type (integer, boolean, real, bottom) of an expression and checks its correctness using type–matching rules. Default value of sys is $result. The expression may be given as a string, a position vector (following the convention of the Mathematica function Position) in the system, or as an AST. expType returns bottom if the expression is not correctly typed. In such a case, the offending tree and the colliding types are printed on the screen.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

4.2.6 getContextDomain

getContextDomain[sys,pos] computes the context domain in which an expression at position pos in sys is used. Default value of sys is $result. The context is the domain inherited by an expression from the constructs which surrounds it. The position is counted from the root of the system following the Mathematica notion of position.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

4.2.7 matchTypes

matchTypes[type1,type2] computes the type corresponding to the union of those of the parameters (integer, boolean, real) and returns the resulting type or bottom if the types are incompatible.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

4.2.8 replaceByEquivExpr

replaceByEquivExpr[sys,pos,expr] replace in sys the expression present at position pos by expr. Default value of sys is $result. replaceByEquivExpr[sys,expr1,expr2] replace in sys all occurences of expr1 by expr2. expr1 and expr2 can be given in AST form of Alpha form. Warning: the equivalence of the resulting program is NOT guaranteed. A test is made to check if both expressions are equivalent, but if this test fails, the replacement is done anyway. This function should be used for replacing an expression by an equivalent one when this cannot be done automatically (for instance, in presence of degenerated domains).
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

4.2.9 setBitWidth

setBitWidth[var:_String,value:_Integer] change the scalar type of variable var1 in system sys if this variable has a specified bitwidth (e.g. integer[S,5]) and set it to the bitwidt value, if the variable does not have a specified bitwidth, it does nothing and issue a warning
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

Chapter 5

Subsystems

This package contains functions to handle Alpha subsystems.

5.1 The Alpha‘SubSystems‘ package

Documentation revised on August 10, 2004

Bugs: test[SubSystems] fails. inlineAll should test that $result contains a system.

5.1.1 SubSystems

The Alpha‘SubSystems‘ package contains functions to operate on Alpha subsystems. These functions are assignParameterValue, assignParameterValueLib, fixParameterValue, inlineAll, inlineSubSystem, spread, removeIdEqus, simplifyUseInputs, substDom, subSystemUsedBy, and topoSort.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.2 addParameterId

function externalized for debugging purposes.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.3 affExtHom

function externalized for debugging purposes.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.4 assignParameterValue

assignParameterValue[param,v,sys] gives the value v to the parameter param in the Alpha system sys and returns the modified system. param is a string, and v an integer. Default value of sys is $result.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.5 assignParameterValueLib

see fixParameter.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.6 fixParameter

fixParameter[param,v,systName] gives the value v to a parameter param in an Alpha system of name ”systName” and in the call to ”systName” in $library. It returns the new library (list of Alpha systems) and modifies $library (except if $library is explicitely specified as the 4th parameter). If the system name is ommited, the function sets the parameter in all the programs of $library and in $result. WARNING: currently, this function supposes that the parameter ”param” has the same value everywhere, you also have to ensure that $result is the top calling system of the library (i.e. the module in an AlpHard program).
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.7 inlineAll

inlineAll[options] inlines all the subsystems of system $result. Options are rename, renameCounter, verbose, caller, library, and current. (see Options[inLineAll] for default values).
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.8 inliningRenameCounter

Variable. Holds the counter value used to avoid name conflicts while renaming the variables.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.9 inlineSubsystem

inlineSubsystem[name,options] searches the current system (see option : caller) for a use statement of the subsystem name, and replaces this use statement with its definition extracted from $library, with proper variable renaming and parameter instanciation. Returns the modified system if no error, the caller otherwise. Side effect: if the "current" option is set, it sets $program to the previous Alpha‘$result and sets $result to the returned value. Options are occurence, rename, renameCounter, verbose, caller, library, undescore, and current (default options in Options[inlineSubsystem]).
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.10 library

library is an option of inlineAll and inlineSubsystem. library → list of systems (default $library) specifies the list of systems to search for a subsystem. When more than one declaration of the same system appear in the library, the first one is inlined.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.11 occurence

Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.12 underscore

option of inlineSubsystem and inlineAll (Boolean). When True (default), new identifier separator is _, whereas it is X otherwise.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.13 spread

spread[var, index] replaces var in system $result by a set of variables varI, where I is in the range of index of var. This function is not completed and should not be used.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.14 removeIdEqus

removeIdEqus[sys] removes in sys the equations of the form A=B as introduced for example by the inlineSubsystem and inlineAll transformations. Warning, this function does not normalize the resulting system, unless option norm → True is used. removeIdEqus has options norm, allLibrary and inputEquations.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.15 simplifyUseInputs

simplifyUseInputs[sys] adds local buffer variables for each input which is not already a simple variable (input to a use may be any expression). The new variable is defined on the 'real domain' of the expression: its context domain intersected with its use domain.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.16 substDom

substDom[dom, extDom, paramAssign, nbparams] computes the transformation of a domain dom in a subsystem inlining. For internal use only.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.17 subSystemUsedBy

subSystemUsedBy[sys] returns the list of names of subsystems used by system sys (default $result).
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

5.1.18 topoSort

topoSort[lib] returns the list of system names of library lib, sorted in the reverse hierachical order, i.e. if a system A uses a system B and a system C, topoSort returns {B, C, A} or {C, B, A}
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

Chapter 6

Manipulating Alpha Expressions

The Matrix and Tables packages contain some functions to manipulate Alpha expressions and programs. Other such functions are scattered in other packages, as they have been written sometimes on the fly by the authors of these packages. A developper has to read this chapter carefully in order not to reinvent the wheel.

6.1 The Alpha‘Matrix‘ package

Documentation revised on March 11, 2008

6.1.1 Matrix

The Alpha‘Matrix‘ package contains a few functions related to Alpha matrices. These functions are alphaToMmaMatrix, composeAffines, convHull, convexize, convexizeAll, deleteColumn, deleteRow, determinant, dropColumns, dropParameters, dropRows, emptyLinearPartQ, getLinearPart, getTranslationVector, hermite, hermiteL, hermiteR, inverseMatrix, identityQ, idLinearPartQ, idMatrix, inverseInContext, nullSpaceVectors, mmaToAlphaMatrix, nullLinearPartQ, simplifyAffines, solveDiophantine, smithNormalForm, squareMatrixQ, suppressRowNum, translationMatrix, translationQ, unimodularQ, composeAffines, inverseMatrix, unimodularQ, unimodularCompletion, unimodCompl.
Defined in file: Alpha/Matrix.m

Package: Alpha‘Matrix‘

6.1.2 addLinSpace

Obsolete function, replaced by the generalized ChangeOfBasis.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.3 alphaToMmaMatrix

alphaToMmaMatrix[mat] translates a linear function from Alpha format into Mathematica format (list of list). If the linear function mat was affine (non nul constant part) this constant part is lost.
Example:
alphaToMmaMatrix[
matrix[3, 3, {i, j}, {{1, 2, 0}, {0, 3, 0}, {0, 0, 1}}]
]
gives {{1, 2}, {0, 3}}.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.4 canonicalProjection

Obsolete. canonicalProjection[dom, pos] returns a projected domain obtained by supressing the indices given by the integer list pos.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

OBSOLETE. Replaced by (generalized) ChangeOfBasis.

6.1.5 composeAffines

composeAffines[m1,m2] returns the composition of two affine matrices m1 and m2. These matrices are described in the Alpha format.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.6 convHull

convHull[ldom] returns the convex hull of a (possibly empty) list of domains ldom.

Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

If the input list is empty, returns an empty list.

6.1.7 convexize

convexize[domain] returns the convex hull of domain if domain is convex, otherwise returns domain unmodified. Used to convexize a union of domains (try to use DomConvex instead).
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.8 convexizeAll

convexizeAll[sys] tries to convexize all the non-convex domains of sys (default $result) and returns the modified system.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.9 deleteColumn

deleteColumn[m,k] removes the k-th column of the Alpha matrix m. It removes also index k of m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.10 deleteRow

deleteRow[m,k] removes the k-th row of the Alpha matrix m. It does not touch the index list of m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.11 determinant

determinant[m] gives the determinant of a square Alpha matrix. The result can be integer or rationnal, the constant part is not taken into account.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.12 dropColumns

dropColumns[m,n] removes n columns of the Alpha matrix m, from front if n is positive, from end if n is negative. This function does not touch the dimensions nor the indexes of the matrix.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.13 dropParameters

dropParameters[sys,m] removes the rows of Alpha matrix m corresponding to parameters of system sys (default $result). If there are too many parameters, dropParameters returns m unchanged.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.14 dropRows

dropRows[m,n] removes n rows of the Alpha matrix m, from front if n is positive, from the end otherwise. This function does not touch the dimensions nor the indexes of the matrix.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.15 emptyLinearPartQ

emptyLinearPartQ[m] is True if the linear part of matrix m is empty, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.16 getLinearPart

getLinearPart[m] returns the linear part of an affine function m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.17 getTranslationVector

getTranslationVector[m] returns the translation vector of a square affine function m.

Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.18 hermite

hermite[m] returns the left hermite decomposition {h,u} (as Alpha matrices) of m1 (such that m1=composeAffines[h,u]).
WARNING: Currently, this function only deals with linear function (the constant part is ignored).
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.19 hermiteL

hermiteL[m] performs the left Hermite decomposition of the Mathematica matrix m and returns mathematica matrices {H,Q} where Q unimodular, H upper triangular and m = Dot[H,Q].
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.20 hermiteR

hermiteR[m] computes the right Hermite decomposition of the Mathematica matrix m and returns Mathematica matrices {H,Q}, where Q is unimodular, H is lower triangular, and m=Dot[Q,H].
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.21 inverseMatrix

inverseMatrix[m] computes and returns the inverse of a square affine matrix m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.22 identityQ

identityQ[m] is True if an Alpha affine function m is the identity, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.23 idLinearPartQ

idLinearPartQ[m] is True if the linear part of the affine function given as Alpha matrix m is the identity, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.24 idMatrix

idMatrix[idx_List,idy_List] returns the transformation matrix for dependence (idx→idy), where idx and idy are lists of index names. It is assumed that names in idy are also in idx.
Example:
idMatrix[{”i”,”j”},{”i”,”i”,”j”}] =
matrix[4, 3, {”i”,”j”}, {{1,0, 0}, {1, 0, 0}, {0, 1, 0}, {0, 0, 1}}].
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.25 inverseInContext

inverseincontext[m,d] computes the inverse matrix of Alpha matrix m in context with domain d, i.e., taking into account the lineality space defined by d.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.26 isIdLinearPart

obsolete form of idLinearPartQ.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.27 isNullLinearPart

obsolete form of nullLinearPartQ.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.28 nullSpaceVectors

nullSpaceVectors[mat] gives the list of vectors of the null space of Alpha matrix mat.

Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.29 mmaToAlphaMatrix

mmaToAlphaMatrix[m] returns the Alpha form of a Mathematica matrix m. mmaToAlphaMatrix[m,i] returns the Alpha form of matrix m with index (list of strings) i. mmaToAlphaMatrix[m,c:List[___]] returns the Alpha form of a linear function Z → mZ+c.
Example:
mmaToAlphaMatrix[{{1, 2},{0, 3}},{1,4}]
gives the Alpha matrix which represents the affine function (i,j→i+2j+1,3j+4).
mmaToAlphaMatrix[m,c,i] returns the Alpha form of linear function Z → mZ+c, with index (string list) i.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.30 nullLinearPartQ

nullLinearPartQ[m] is True if the linear part of the affine function given as Alpha matrix m is null, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.31 solveDiophantine

solveDiophantine[a,b] solves the linear diophantine system of equations aX = b where a is a MMA matrix, and b a MMA vector. The solution has the form {x1,n,M} where x1 is a particular solution of aX=b, n is the number of columns of M, and M is a matrix such that X=Mt (t is a n-vector) is the general solution of aX=0. The general solution of aX=b is x1+Mt. If the system has no integral solution, then x1 is {}. solveDiophantine[a] solves the linear diophantine system of equation ax=0 where a is a MMA matrix.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.32 smithNormalForm

smithNormalForm[mat computes the Smith Normal Form of an Alpha matrix mat and returns {u,s,v} (Alpha matrices), where u,v are unimodular and s is diagonal such that mat = u.s.v. smithNormalForm[m] computes

the Smith Normal Form of a Mathematica matrix m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.33 squareMatrixQ

squareMatrixQ[m] is True if m is a square Alpha matrix.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.34 subMatrices

subMatrices[m1,m2] subtracts two matrices m1 and m2. This function is for internal use by the Pipeline function.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.35 suppressRowNum

suppressRowNum[mat,i] suppresses row i in (MMALPHA) matrix mat.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.36 translationMatrix

translationMatrix[ind, vec] returns an Alpha translation matrix corresponding to the function: (ind → ind+vec), where ind is a list of indices and vec an integral vector.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.37 translationQ

translationQ[m] is True if the full rank affine function m is a translation, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.38 unimodularQ

unimodularQ[m] is True if an MMALPHA affine matrix m is square and unimodular, False otherwise.

Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.39 unimodularCompletion

unimodularCompletion[vl] completes a vector list vl into a unimodular matrix which is returned. unimodularCompletion[mat] takes the vector expressed as an Alpha matrix mat and completes it into a unimodular Alpha matrix which is returned. If the original vector is of size n, the resulting matrix is of size (n+1)*(n+1) and the first row of the resulting matrix is the original vector with 0 appended as constant term. For example,
unimodularCompletion[{1,1,1}] returns
matrix[4, 4, {}, {{1, 1, 1, 0}, {0, 0, 0, 1}, {0, 0, 1, 0}, {0, 1, 0, 0}}]
and
unimodularCompletion[matrix[4, 4, {}, {{1, 1, 1, 0}, {0, 0, 0, 1}}]
returns
matrix[4, 4, {}, {{1, 1, 1, 0}, {0, 0, 0, 1}, {0, 0, 1, 0},
{0, 1, 0, 0}}].
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.40 unimodCompl

unimodCompl[mat] returns an unimodular completion of matrix mat, or $Failed if an error occurs. The result is an Alpha matrix. The indices of the result are those of mat, with some new arbitrary indices if necessary. Exemple : completion of (i,j→2i) returns (i,j,k→2i+k,j,i). The optional parameter indicates the dimension of the parameter space (0 if none).
unimodCompl[mat] is the mathematica version (no constant row and column). It returns a square unimodular MMALPHA matrix. First lines of this matrix are the matrix mat (modulo an extension on the right).
For exemple: {{2,0,0},{1,2,0}} will be completed as:
{{2,0,0,1,0}, {1,2,0,0,1}, {0,0,1,0,0}, {1,0,0,0,0}, {0,1,0,0,0}}.
unimodCompl[mat,nbPar] returns a MMA matrix or $Failed if an error occurs. mat is a (k*(n+p+1) MMA matrix, it represents a k-dimensional affine function for a n-dim variable, with a p-dim parameter space. Thus, the last column is the constant column. nbPar is the parameter space dimension, i.e. p. The function computes n from p and mat. The result is a MMA matrix, which has at least (n+p+1) rows and exactly (n+p+1) columns. The first k rows are exactly mat. If the result isn't square, it means that

some dimension will be added by the returned change of basis.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.1.41 simplifyAffines

simplifyAffines[] simplifies all affine functions in system $result.
simplifyAffines[sys] simplifies all affine functions on system sys.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

6.2 The Alpha‘Tables‘ package

Documentation revised on March 11, 2008

6.2.1 Tables

The Alpha‘Tables‘ package contains variable and function definitions relative to the retrieval of contextual information on nodes and programs. Functions are: addAllParameterDomain, addParameterDomain, getDeclaration, getDeclarationDomain, getDefinition, getDimension, getEquation, getIndexNames, getInputVars, getLocalVars, getOutputVars, getVariables, getSystemName, getSystemParameters, getSystemParameterDomain, symToString, lookUpFor, lookUpForPos, accessByPath, and undoModif.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.2 addAllParameterDomain

addAllParameterDomain[sys] adds the constraints present in the parameter domain of system sys to each domain of the system. addAllParameterDomain[] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.3 addParameterDomain

addParameterDomain[dom, paramdom] adds to the constraints of dom the constraints on the parameters defined in paramdom. For internal use mostly.

Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.4 getDeclaration

getDeclaration[var] returns the complete declaration of variable var in system $result or an empty list if the declaration does not exist. getDeclaration[sys, var] returns the declaration of var in system sys.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.5 getDeclarationDomain

getDeclarationDomain[v] returns the declaration domain of variable v in system $result. getDeclarationDomain[sys,v] returns the declaration domain of variable v in system sys
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.6 getDefinition

getDefinition[var] returns the RHS of the equation defining variable var or an empty list if the definition does not exist. If the variable is defined as output of a subsystem, returns an empty list and prints an error message.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.7 getDimension

getDimension[dom] returns the dimension of the Alpha domain dom.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.8 getEquation

getEquation[var] returns the equation defining a variable var or an empty list if the definition does not exist. If the variable is the output of a subsystem, getEquation issues a warning and returns the use statement of this subsystem.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.9 getIndexNames

getIndexNames[sys, var] returns the list of index names from a variable's declaration. getIndexNames[var] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.10 getInputVars

getInputVars[sys] returns the list of input variables used in system sys. getInputVars[] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.11 getLocalVars

getLocalVars[sys] returns the list of local variables used in system sys. getLocalVars[] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.12 getOutputVars

getOutputVars[sys] returns the list of Output variables used in system sys. getOutputVars[] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.13 getVariables

getVariables[sys] returns the list of variables used in system sys. getVariables[] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.14 getSystemName

getSystemName[sys] return the name of system sys. getSystemName[] gives the name of $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.15 getSystemParameters

getSystemParameters[sys] return the parameters of the system sys (default $result)
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.16 getSystemParameterDomain

getSystemParameterDomain[sys] return the parameters domain of the system sys (default $result)
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.17 symToString

symToString[symb_] Converts a symbol to string. Leaves any other object simply evaluated. Returns the string containing its name according in current output format.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.18 lookUpFor

lookUpFor[sys, position:{_Integer...}, operator_Symbol] returns the smallest tree whose root is the specified by operator and which contains the specified position, or an empty list if operator not found.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.19 lookUpForPos

lookUpForPos[sys, position:{_Integer...}, operator_Symbol] returns the position of the nearest occurrence of a specific operator surrounding a specified position, or an empty list if operator not found.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

6.2.20 accessByPath

accessByPath[path:{_Integer..}] extracts the part of $result specified by subtree position path. accessByPath[tree_, path:{_Integer..}] extracts the part

of a tree specified by its subtree position specifier path
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

Only one position can be supplied at a time, so a single tree is returned if the position is valid (no check is done.)

6.2.21 undoModif

undoModif[] called after a program transformation undoes its effects. Returns the program as before the last transformation.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

’undoModif’ copies the contents of ’Alpha‘$program’ into ’Alpha‘$result’ so the result of the last transformation (see description of ’Alpha‘$result’ and ’Alpha‘$program’) is reset to the result of the previous one.

Chapter 7

Elementary Program Transformations

7.1 The Alpha‘ChangeOfBasis‘ package

Documentation revised on August 3, 2004

7.1.1 ChangeOfBasis

The Alpha‘Package‘ package contains the definition of the ”change of basis” transformation (see changeOfBasis).
Defined in file: Alpha/ChangeOfBasis.m
Package: Alpha‘ChangeOfBasis‘

7.1.2 changeIndexes

changeIndexes[sys, var, rules] replaces all indexes in the definition of var according to replacement rules. changeIndexes[var, rules] does the same on $result. Remember that var names are strings. The declaration of var is modified according to the rules. If var is the output of a use statement, all outputs of the use and the use itself are changed.
Defined in file: Alpha/ChangeOfBasis.m
Package: Alpha‘ChangeOfBasis‘

7.1.3 changeOfBasis

changeOfBasis[var.fn] returns a copy of $result in which the variable var is reindexed by an unimodular affine function fn. The change of basis is specified as "B.(i,j,k → i+1,j,k)" where B is the variable to be reindexed, and the change of basis matrix has the format of an Alpha dependence, with one difference: the index mapping function is understood as a mapping from initial to new position (and not the opposite). If it is square, this matrix should be unimodular. Non-square change of basis (also called generalized change of basis) are allowed, in that case, the function should be called this way: changeOfBasis[var.fn, {index, ...}] applies a change of basis fn on variable var, except that the new indices of var are named according to the second argument.
Example:
changeOfBasis["B.(i,j → 1,i,j)",{"i1","j1","k1"}]).
changeOfBasis[sys,var.fn, {index, ...}] applies a change of basis to system sys instead of $result.
changeOfBasis["B.(i,j → 1,i,j)", recurse→True] recursively executes the change of basis on subsystem 'subsys' called with B as input or output. The recursive change of basis has many restriction: as the semantics of the subsystem is modified, we impose that there is only one occurence of subsys appearing in the system. Moreover, the recursive change of basis can modify only local indices (indices which are not extension indices in the use of subsys) and must involve only local indices and parameter transmitted to the subsystem. The change of basis on the extension indices can be performed with the function extDomainCOB[] (see ?extDomainCOB).
Defined in file: Alpha/ChangeOfBasis.m
Package: Alpha‘ChangeOfBasis‘

7.1.4 extDomainCOB

extDomainCOB["subSysName.(j→g(j,N))"] applies the change of basis to all the variables implied in the use of the subsystem subSysName and change the extension domain accordingly. This change of basis should be applied to a system where the following use appear:
use {j| ....} subSysName[...] (...) returns (...)
j are the extension indices, N are the parameter of the caller. The change of basis must only depend on the extention indices and on the parameters as indicated here. The parameter assignement function must be simple (i.e. each parameter of the subsystem is assigned to a parameter of the

caller, e.g. (N,M,P→P,N). This transformation should have no impact on the subsystems.
Defined in file: Alpha/ChangeOfBasis.m
Package: Alpha‘ChangeOfBasis‘

7.2 The Alpha‘Cut‘ package

Documentation revised on August 3, 2004.

7.2.1 Cut

Alpha‘Cut‘ is the package which contains the cut, decompose and merge transformations.
Defined in file: Alpha/Cut.m
Package: Alpha‘Cut‘

7.2.2 cut

cut[var_String, dom_String, outvar1_Sting, outvar2_String] returns the program obtained by cutting the definition of var in program $result into two definitions outvar1 and outvar2, where outvar1 is restricted to dom, and outvar2 is restricted to the complemtary space of dom. The result of cut is put in symbol $result. cut[sys, var, dom, outvar1, outvar] does the same to program sys. The parameter dom may be specified either as a string, or as the internal Alpha form of a domain.
Example: cut["A","{i,j | i<j }","A1", "A2"].
Defined in file: Alpha/Cut.m
Package: Alpha‘Cut‘

7.2.3 decompose

decompose[expr,name] returns the program obtained by adding to $result a new equation "name = expr" and replacing the first occurrence of expr by name. The program returned is assigned to $result. decompose[sys,expr,name] does the same to program sys, but does not modify $result. expr and name are either strings or ast's. expr can also be specified using a Mathematica Position.
Defined in file: Alpha/Cut.m
Package: Alpha‘Cut‘

7.2.4 exprLocalEquivQ

Predicate. Checks whether or not two expressions in an ALPHA programs are equivalent in any point of a given domain. Obsolete function.
Defined in file: Alpha/Cut.m
Package: Alpha‘Cut‘

7.2.5 merge

merge[var1, var2, newVar] replaces in $result two local variable definitions with a single one. The declaration domain of the new variable is the union of the domains of the old ones. The definition of the new variable is a case whose branches are definitions of the old variables restricted to their respective domains. RHS occurrences of the old variables are replaced by the proper restrictions of the new variable.
merge[sys, var1, var2, newVar] replaces in system sys two local variable definitions with a single one and returns the new system.
Defined in file: Alpha/Cut.m
Package: Alpha‘Cut‘

7.2.6 mergeCaseBranches

mergeCaseBranches[casePosition_List, branchPosition_List], merges (in $result) the specified case branches if they contain identical expressions. mergeCaseBranches[sys, casePosition, branchPosition] merges the specified case branches of system sys if they contain identical expressions and returns the new system. The domain of the new branch may be non-convex.
Defined in file: Alpha/Cut.m
Package: Alpha‘Cut‘

7.2.7 mergeIdCaseBranches

mergeIdCaseBranches[sys] tries to merge case branches that have identical expressions (i.e. tries to inverse what splitCaseUnion did). Warning, this function works only on normalized programs.
Defined in file: Alpha/Cut.m
Package: Alpha‘Cut‘

7.2.8 splitCaseUnion

splitCaseUnion[sys] splits all the case branches that have a non convex domain as restrict (union of convex) by several branches, each corresponding

to one convex domain. Warning, this function works only on normalized programs.
Defined in file: Alpha/Cut.m
Package: Alpha‘Cut‘

7.2.9 unionMerge

unionMerge[sys, firstVar, secondVar, resultingVar] is an extension of merge that handles overlapping definitions parameters. Returns the modified program. Warning obsolete function.
Defined in file: Alpha/Cut.m
Package: Alpha‘Cut‘

unionMerge acts much like merge, the difference being that overlapping domains are accepted if both expressions are equivalent over the overlap. By equivalent we mean that the structures of the expressions are identical and that the images of the overlap domain through the dependence functions are equal for any dependence. To illustrate the latter, consider the expressions

{i,j|i=j; 1<=j<=4
} : A.(i,j→i,j)

and

{i,j|i=j; 1<=j<=4} : A.(i,j→j,j).

Although the dependence functions are not equal, these expressions are equivalent over the domain {i,j|i=j; 1<=j<=4}.

7.3 The Alpha‘Normalization‘ package

Documentation revised on August 1, 2004

7.3.1 Normalization

Alpha‘Normalization‘ is the package containing the definitions of normalization rules along with basic normalizing functions: checkRestrictions, minRestrictInCtxt, normalize, normalize0, normalizeDef, normalizeDef0, normalizeInCtxt, normalizeInCtxt0, normalizeQ, normalize0Q, and simplifyInContext.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

7.3.2 checkRestrictions

checkRestrictions[ast] checks for redundant restrictions in a normalized system. Default ast is Alpha‘$result.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

This function is useful after normalizing a complete system, in order to remove redundant restrictions. It cannot be included in the set normalization rules because it needs contextual information wrt. the normalized expression (it requires the declaration of a variable).

7.3.3 minRestrictInCtxt

minRestrictInCtxt[ast,domain] simplifies all restrictions of a program in the context of a specific domain. Returns the simplified expression.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

7.3.4 normalize

normalize[option] normalizes program $result with the given option. normalize[sys,option] normalizes ALPHA expression sys with the given option. The available option is indexnorm which should be set to True if normalization of index expressions is desired. The default option is False.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

normalize[exp] computes the fixpoint of a set of normalization rules when applied to exp. The set of rules is contained in variable normalizationRules.

7.3.5 normalize0

normalize0[ast] normalizes an ALPHA expression ast wrt. Alpha0 rules (allowing nested cases.) Default ast is Alpha‘$result.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

normalize[exp] computes the fixpoint of a set of normalization rules when applied to exp. The set of rules is contained in variable normalizationRules0.

7.3.6 normalizeDef

normalizeDef[symbol] normalizes the definition of ALPHA variable symbol in $result. normalizeDef[sys,symbol] normalizes the definition of ALPHA variable symbol in program sys. Variable symbol is a string.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

normalizeDef[exp, var] computes the fixpoint of a set of normalization rules when applied to the definition of var in exp. The set of rules is contained in variable normalizationRules.

7.3.7 normalizeDef0

normalizeDef0[symbol] normalizes the definition of ALPHA variable symbol in $result wrt. the rules of Alpha0 (allowing nested cases). normalizeDef0[sys,symbol]normalizes the definition of ALPHA variable symbol in program sys. Variable symbol is a string.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

normalizeDef[exp, var] computes the fixpoint of a set of normalization rules when applied to the definition of var in exp. The set of rules is contained in variable normalizationRules0.

7.3.8 normalizationRules

Set (list) of rewrite rules specifying the normalization of an ALPHA expression towards the CRD (Case-Restriction-Dependence) normal form.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

normalizationRules can be extended by appending/prepending new rules to the initial list.

The rules are named and have the form name = exp1 :> exp2.

7.3.9 normalizationRules0

Set (list) of rewrite rules specifying the normalization of an ALPHA expression towards the Alpha0 (NestedCase-Restriction-Dependence) normal form.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

normalizationRules0 can be extended by modifying the value of the rule list. The rules are named and have the form name = exp1 :> exp2.

7.3.10 normalizeInCtxt

normalizeInCtxt[ast,domain] returns a normalized and reduced form of the restriction of ast to domain.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

'normalizeInCtxt' should perhaps operate in a more intelligent way.

7.3.11 normalizeInCtxt0

Function. Normalizes an expression in the context of a specific domain.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

normalizeInCtxt0 should perhaps operate in a more intelligent way.

7.3.12 normalizeQ

normalizeQ[ast] returns True if ast is normalized, False otherwise. Default value for ast is Alpha‘$result.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

7.3.13 normalize0Q

normalize0Q[ast] returns True if ast is normalized, False otherwise. Default value for ast is Alpha‘$result.

Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

7.3.14 correctMat

in test
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

7.3.15 correctAffineFunctions

in test 2
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

7.3.16 simplifyInContext

simplifyInContext[] simplifies and reduces the Alpha system $result in the following ways: simplifies restriction domains in context of declaration, simplifies transformations in context of declaration. It operates on normalized or unnormalized programs. simplifyInContext[sys] simplifies and reduces the Alpha system sys and returns the simplified system. Does not change $result. simplifyInContext[sys, exp] simplifies expression exp in the context of sys and returns the simplified expression.
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

7.3.17 simplifySystem

simplifySystem[] is just a shorthand for simplifyInContext[]; convexizeAll[];normalize[] (or normalize0 depending on the option). simplifySystem[sys] return system sys modified without modifying $result
Defined in file: Alpha/Normalization.m
Package: Alpha‘Normalization‘

7.4 The Alpha‘Reduction‘ package

Documentation revised on August 8, 2004

7.4.1 Reduction

The Alpha‘Reduction‘ package contains transformations to deal with reduction operators.
Defined in file: Alpha/Reduction.m
Package: Alpha‘Reduction‘

This package is not completed. In particular, it contains two functions splitReduction and splitReduce which do no work.

7.4.2 serializeReduce

serializeRecuce[pos] serializes the reduction at position pos in $result. serializeReduce[pos, spec] serializes the reduction at position pos in $result using serialization specifier spec (spec is a string). Parameter pos may be either a position in the AST, or the name (string) of a variable whose definition has the form "pos = reduction". serializeReduce[sys, pos, spec] does the same to program sys. Parameter spec is a string containing the new variable name composed with the new serialized dependence (parameters need not be given explicitly). For example: "Z.(i,k→i,k-1)" means that the serialized variable will be "Z" and the serialize direction will be the vector (0,-1). If no spec is given, the function guesses the direction of serialization, and the option (invert → False|True) allows this direction to be changed. The guess is based on the null space of the reduction function, and it works only if this null space has exactly one vector.
Defined in file: Alpha/Reduction.m
Package: Alpha‘Reduction‘

7.4.3 isolateReductions

isolateReductions[] locates all the reductions in a system and isolates these reductions in new equations
Defined in file: Alpha/Reduction.m
Package: Alpha‘Reduction‘

7.4.4 isolateOneReduction

isolates one reduction
Defined in file: Alpha/Reduction.m
Package: Alpha‘Reduction‘

7.4.5 splitReduction

splitReduction[y] tries to rewrite the reduction at rhs of symbol y as a sequence of reduction. This function does not work currently.
Defined in file: Alpha/Reduction.m
Package: Alpha‘Reduction‘

7.5 The Alpha‘Substitution‘ package

Documentation revised on August 8, 2004

7.5.1 Substitution

Alpha‘Substitution‘ is the package which contains the functions for substituting Alpha variables.
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

7.5.2 addLocal

see addlocal in Substitution.m
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

7.5.3 addlocal

addlocal[sys, var, exp] declares a new local variable var in system sys and defines it as exp on the domain of exp as calculated by expDomain. All instances of exp in sys are replaced with the variable var.
addlocal[sys, var, pos] adds a new local variable var defined on the context domain of pos (as calculated by getContextDomain[]) and adds the definition of var as the expression at position pos. The expression defined by pos is replaced by var. Default value of sys is $result. WARNING: This function is kept for backward compatibility, but it is not sufficiently specified, please use rather addLocalLHS or addLocalRHS functions
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

7.5.4 addLocalLHS

addLocalRHS[var, exp] is similar to addlocal (see addlocal) but its action is more clearly defined. From a set of equations such as
A =...; X =...A...
addLocalLHS["B","A"] changes it into
B = A; A =...; X =...B....
i.e., B is added in the LHS of the equation using A.
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

7.5.5 addLocalRHS

addLocalRHS[var, exp] is similar to addlocal (see addlocal) but its action is more clearly defined. From a set of equations such as
A =Y; X =...A...
addLocalRHS["B","A"] changes it into
A=B;B=Y;X=...B....,
i.e., B is added in the RHS of the equation using A, except if A is an input, in which case it acts as addLocalLHS:
B=A; X=...B....
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

addlocal is more or less the inverse transformation of substituteInDef.

7.5.6 getNewName

getNewName[sys,var] checks that the identifier "var" is not already used in sys (default $result) and returns it if not. If this identifier already exists, it returns a modified version of "var" by duplicating its last letter.
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

7.5.7 getOccurs

getOccurs[sys,p] finds out the positions of a pattern p in system sys, and returns a Mathematica position specifier containing the list of occurrences of p in $result. The pattern p can be a string (e.g. "A.(i,j→i+j)"), in which case it is parsed, or it can be an Alpha AST. The result of getOccurs is a list of positions specifiers that are defined with respect to sys. A position

specifier can then be used in the function Part to access the element. For example, if the position is {6,2,3}, then Part[sys,6,2,3] gives the element. One can also use getPart[sys,{6,2,3}], which is easier to use directly.
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

7.5.8 getOccursInDef

getOccursInDef[sys, var, p] lists the positions of occurrences of a pattern p in the definition of variable lhs inside an ALPHA program sys (default $result).
getOccursInDef[sys, var, p, rank] gives the position number rank in the result of getOccursInDef[sys, var, p]. rank specifies which occurrence to report (1 = first, 2 = second, {1,2}= first and second, etc).
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

7.5.9 replaceDefinition

replaceDefinition[lhs, rhs] replaces the definition of a variable lhs in an equation of program $result with the Alpha expression rhs. replaceDefinition[sys, lhs, rhs] replaces the definition of variable lhs in an ALPHA program sys with the Alpha expression rhs and return the new system. lhs is a variable name (either symbol or string). rhs is either an ast or a string.
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

The function returns a copy of the original program in which the rhs of the equation defining the specified variable is replaced by an expression passed as parameter. The result of the substitution is not normalized. The meaning of the program may be changed by this transformation.

7.5.10 substituteInDef

substituteInDef[lhs, var] substitutes in $result all occurrences of variable ‘var’ in the RHS of the definition of variable ‘lhs’ by the definition of ‘var’, and returns the new system in $result. substituteInDef[lhs,var,rank] substitutes occurrences ‘rank’ of variable ‘var’ in the RHS of the defition of variable ‘lhs’ by the definition of ‘var’. The parameter rank specifies which occurrence to replace (1 = first, 2 = second, {1,2}= first and second, etc).

substitute[sys,lhs,var] and substitute[sys,lhs,var,rank] do the same on program contained in symbol 'sys'.
Defined in file: Alpha/Substitution.m
Package: Alpha'Substitution'

7.5.11 occursInDefQ

occursInDefQ[sys,var,p] returns True if the pattern p occurs in the of the definition of the variable var. Default value of sys is $result. p can be either a string or an AST.
Defined in file: Alpha/Substitution.m
Package: Alpha'Substitution'

7.5.12 unusedVarQ

unusedVarQ[sys,var] is True if var is used in the rhs of an equation of system sys, False otherwise. Default value of sys is $result.
Defined in file: Alpha/Substitution.m
Package: Alpha'Substitution'

7.5.13 removeUnusedVar

removeUnusedVar[sys,var] removes the definition of unused variable var in system sys. The default value of sys is $result.
Defined in file: Alpha/Substitution.m
Package: Alpha'Substitution'

7.5.14 removeAllUnusedVars

removeAllUnusedVars[sys] removes the definitions of all unused local variables of system sys (default, $result).
Defined in file: Alpha/Substitution.m
Package: Alpha'Substitution'

7.5.15 isOutputRegular

isOutputRegular[sys,o] is True if output variable o has the form o = v where v is a simple variable. This predicate allows one to detect non regular outputs
Defined in file: Alpha/Substitution.m
Package: Alpha'Substitution'

7.5.16 areAllOutputsRegular

areAllOutputsRegular[sys] is True if all outputs of sys have the form o = v, False otherwise. areAllOutputsRegular[] does the same to $result.
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

7.5.17 mkOutputRegular

mkOutputRegular[sys,o] makes output variable o regular, if necessary. In other words, it makes sure that the definition of o has the form o = v where v is a simple local variable. mkOutputRegular[o] does the same to $result.
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

7.5.18 mkAllOutputsRegular

mkAllOutputsRegular[sys] makes all output variables of sys regular. mkAll-OutputsRegular[] does the same on $result.
Defined in file: Alpha/Substitution.m
Package: Alpha‘Substitution‘

Chapter 8

Scheduling

8.1 The Alpha‘ScheduleTools‘ package

8.1.1 applySchedule

applySchedule[] applies the schedule $schedule to $result. The resulting system is the original one where all the change of basis in $schedule have been performed and where the first index can interpreted as the time everywhere except for the input and output variables. applySchedule[sys_Alpha‘system] applies the schedule $schedule to system ‘sys’. applySchedule[sys_Alpha‘system, sched_Alpha‘Schedule] applies the schedult ‘sched’ to system ‘sys’. See also: schedule, $schedule.
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.2 showSchedResult

showSchedResult[sched1_Alpha‘ScheduleResult] pretty prints the schedule sched1, equivalent to show[sched1]
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.3 other

other is an option of appSched which means for other variables
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.4 renameIndices

renameIndices[sys:Alpha‘system,indiceList:{___String}] rename the indices of all the local variables of the system ‘sys’ according to the list given ‘indiceList’ (including the output of ‘use’). renameIndices[sys:Alpha‘system,indiceList:{___String},var:(
rename the change the name in only one variable ‘var’ (or a list of given variable)
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.5 appSched

appSched[sys, sch, options] returns a scheduled system, according to sch. appSched[sch, options] applies schedule sch to $result. appSched[options] applies schedule $schedule to $result. appSched is equivalent to ApplySchedule but works for multidimensionnal scheduling. The projection direction is chosen automatically by MMALPHA, but if necessary, it can be chosen using the projMatrix or the projVector options. See ?appSchedOptions for more informations.
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.6 appSchedOptions

appSched chooses automatically the projection direction for variables, and this sometimes may not be convenient. There are several ways to specify more precisely the schedule. Option projVector allows a projection vector to be specified. For example, projVector → {1,0,0} would say that the first component of all variables will be replaced by the time given by the schedule, in a 3-dimensional space. However, this option can be used only if all variables of the system are already placed in the same space, which may not be the case. Similarly, one can specify the projection matrix, using the projMatrix option. For example, projMatrix → {{1,0,0},{0,0,1}} will project all variables on components 1 and 3. Again, this may not be convenient if all variables are not in the same space. Another way of specifying the projection is to use the timeDimensions option. Specifying timeDimensions → {2} says that the component of all variables that will be replaced by the time is the second dimension. This is equivalent to the option projVector → {0,1,0}, but less dependant on the dimension of the variables. Moreover, one can specify the time dimensions variable by variable, using the variables option. This option may take several forms. variables → varlist,

where varlist is a list of local variables, restricts appSched to be applied to only variables in varlist. variables → {{ var, timeDimensions }, ... } allows the value of option timeDimensions to be attached to each individual variable. For example, variables → {{ "A", {2} }, { "B", {1} }} says that for variable A, the time dimensions is 2, and for variable B, it is 1. One can also factorize this list as in variables →{{{"A","C"}, {2}, {"B", {1}}}, and also use the keyword other to specify the time dimensions for all remaining variables, as in variables → {{ "A", {2} }, { "B", {1} }, { other, {3} }}. Notice that timeDimensions is a list of dimensions: this is a provision for a multi-dimensional schedule, but this has not been tested yet. Also, remember that no garantee is given that these options provide a possible projection, as this depends on the existence of a unimodular transformation combining the schedule and the projection. In other words, appSched may fail sometimes. A final warning: projection vector or matrices should include the parameters' dimensions.
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.7 timeDimensions

timeDimensions is an option of appSched and appVarSched that gives the list of dimensions that are to be assigned to time. This option gives a hint to appSched. Say a variable has indexes i,j,k (including parameters), then timeDimensions→{2} means that index j is going to be replaced by the time and thus, indexes i and k will become processors. This hint may not always work. See also ?appSchedOptions.
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.8 variables

option of appSched. It gives the list of variables for which the schedule must be applied. Default value is Null, and means all variables. See also ?appSchedOptions for the exact syntax of this option.
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.9 appVarSched

appVarSched[sys:_system,var_String,time_Matrix,options] apply a single COB to a variable, the schedule is to be given as a Mathematica matrix (List of

List of integer). This function is for internal use
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.10 convertSchedule

convertSchedule[sch:_scheduleResult]. Takes a schedule table and returns an equivalent time functions table. This format is used by lifetime, appSched, and other functions
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.11 identitySchedule

identitySchedule[sys,dim] build the identity schedule of dimension ‘dim‘ dor system sys. The identity schedule simply consist of the lexicographic order on the ‘dim‘first indices (i.e. the program is already scheduled
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.12 isScheduledQ

isScheduleQ[sys,dim] check whether the sys is scheduled or not (with a schedule of dimension dim. isScheduledQ[sys] assumes a linear schedule
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.13 testSched

testSched[sys:_system, sch:_scheduleResult] tests the the schedule for the given system. When flow dependencies are not satisfied, it prints messages and returns False. testSched[sys:_system, sch:_scheduleResult, var:_String] test schedule for var only. Param sys may be a dependencies table. Default: testSched[] test $result with $schedule.
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.14 equationOrderQ

equationOrderQ[sys] checks if the equations are correctly ordered to generate code with a schedule with duration of 0 for some equation (i.e. graph of

dependencies internal to te loop nest is topologicaly sorted; If the answer is Yes, the function returns True otherwise it attempts to find a permutation of the equations that is correct(returns a list of integer). This function does not take into account dependencies involving input or output variables WARNING, this function is temporarily disable because of an internal bug (TopologicalSort)
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.15 reorderEquations

reorderEquations[sys,permutation] reorder the equation of system sys according to the permutation given (which only concern the local variables, see the function equationOrderQ). The output equation are put at the end of the program.
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.16 buildPseudoAlpha

buildPseudoAlpha[sys_] returns a system ‘sys2’ in which all the subsystems called by ‘sys’ have been replaced by equations on the output variables of the sub-system use. This function is used in order to be able schedule a structured system. buildPseudoAlpha[] is by default applied to $result
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.17 buildSchedConstraints

buildSchedConstraints[sys_,List[sched_scheduleResult]] builds a list of constraints (strings) that correspond to the schedules given as second argument for all the subsystem called in ‘sys’. buildSchedConstraints[] calls the function buildSchedConstraints[$result,$scheduleLibrary]
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.18 buildSchedConstraintForUse

buildSchedConstraintForUse[sys_,use1_use,sched_SchedulResult] build a list of constraints (strings) that correspond to the schedule given in sched of the subsystem used in ‘use1’ which is part of ‘sys’

Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.19 isReducibleQ

isReducibleQ[sys] checks wether a structured Alpha program is ‘reducible’ in the sens described in Irisa research report PI1140: the dependency between output and input of a subsystems have identity on the extension part
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.1.20 addBufferVars

addBufferVars[sys:_Alpha‘system] adds systematically a local ‘buffer’ variables for each input and each output of each use of subsystems in ‘sys’ (warning, if the input is not a simple variable, the function fails, this function is to be for internal use, please refer to simplifyUseInputs[])
Defined in file: Alpha/ScheduleTools.m
Package: Alpha‘ScheduleTools‘

8.2 The Alpha‘Schedule‘ package

8.2.1 Schedule

Package. Schedule for alpha program The only function used is : schedule[]
Defined in file: Alpha/Schedule.m
Package: Alpha‘Schedule‘

8.2.2 structSched

structSched[] tries to find a structured scheduling for $result, using $scheduleLibrary for schedule of subsystems. The default mode is linear, i.e. no dimension is added for subsystems. By setting the option structSchedType to multi, the schedule of the subsystems will be in an additionnal dimension. Warning, in that case, you MUST set the depth of the schedule expected with the multSchedDepth options. Example: structSched[structSchedType→multi,multiSchedDept
finds the schedule of $result and puts
Defined in file: Alpha/Schedule.m
Package: Alpha‘Schedule‘

8.2.3 schedule

schedule[] finds the schedule of $result and puts the result in $schedule. schedule[sys_Alpha‘system], finds the schedule of alpha system ‘sys’ and puts the result in $schedule (affine by variable by default). schedule[sys_Alpha‘system, options] calls schedule with non default options. Options[schedule] provides the options of schedule, and ?opt1 provides information on option opt1. ?$schedule provides info on the output format of the schedule. The schedule computation may take a long time (2 minutes for 20 instructions). More information is available in the file $MMALPHA/doc/user/¡=Scheduler_user_manual.ps.
Defined in file: Alpha/Schedule.m
Package: Alpha‘Schedule‘

8.2.4 checkOptions

checkOption[sys_Alpha‘system,options___Rule] check that the set of option of schedule is coherent, returned a new set of option in which some option have been changed by default to cope with incoherent options if possible
Defined in file: Alpha/Schedule.m
Package: Alpha‘Schedule‘

8.2.5 benchSched

benchSched[f] runs the scheduler on alpha file f, or on list of alpha files f.
Defined in file: Alpha/Schedule.m
Package: Alpha‘Schedule‘

8.2.6 farkas

option of benchSched
Defined in file: Alpha/Schedule.m
Package: Alpha‘Schedule‘

8.2.7 vertex

option of benchSched
Defined in file: Alpha/Schedule.m
Package: Alpha‘Schedule‘

8.3 The Alpha‘VertexSchedule‘ package

8.3.1 Recent modifications (as of Dec. 2004)

An additional note regarding extensions of `VertexSchedule.m` to handle multi-rate dataflow systems is available in file

`$MMALPHA/doc/referenceManual/dataflow.tex`

Documentation revised on August 1st, 2004.

8.3.2 VertexSchedule

The Alpha‘VertexSchedule‘ package is one of the two scheduling packages of MMALPHA. Functions of the Alpha‘Schedule‘ package should be preferred, but in some cases, VertexSchedule offers some interesting functionalities. Symbols defined in this package are $dependencyConstraints, $optimalityConstraints, $scheduleDepTable, $scheduleLibrary, $scheduleMDSol. Functions are checkSchedOptions, constOf, depComponents, depCycles, depGraph, DomDimension, DomSingletonQ, DomTrueDimension, eliminateVar, encodeSchedule, equations, ishow, isIndentityOnDim, getLinPart, getUseSchedule, loadScheduleLibrary, matrixTransPart, polDomainQ, timeMinSchedConstraints, saveScheduleLibrary, scd, slackOf, sortEquations, zpolDomainQ
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.3 $dependencyConstraints

$dependencyConstraints contains the dependency constraints used for solving the scheduling. It is a list whose entries correspond directly to the entries of the deptable. The last element is the set of dependency variables. This global variable is set when calling the scheduleConstraints function
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.4 $optimalityConstraints

$optimalityConstraints contains the optimality (total time, etc) constraints used for solving the scheduling. It is a list, whose entries are pairs {variables, constraints}. This global variable is set by the timeMinSchedConstraints function

Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.5 $scheduleDepTable

when set, $scheduleDepTable containts the dependency table of $result. It is computed by a call to scd.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.6 $scheduleLibrary

$scheduleLibrary containts the schedules found by the scheduler when called on the elements of the current library. These schedules are used when the scheduler is called with the structured option
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.7 $scheduleMDSol

$scheduleMDSol is an array which gathers information during a call to the scd function. $scheduleMDSol[i] contains a list {o,c,v,sol} consisting of the objective function, constraints, variables, and solution to the linear problem solved when dealing with dimension i of the schedule. For a unidimensional schedule, only level 1 is set. For a dataflow schedule, level 1 corresponds to the dataflow leve, and other levels to the multidimensional schedule of the inside indexes.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.8 adaptUses

adaptUses[] modifies all use statements, by adding extra input equations
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.9 listOfAdaptedSignals

listOfAdaptedSignals is an option of adaptUses, False by default. If True, adaptUses returns a list formed of the equivalence between the adapted signals and the old signals

Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.10 addSchedule

addSchedule is an option of loadScheduleLibrary. Default value is True. If True, the schedule contained in file sys.scdlib is added to $scheduleLibrary, where it replaces a previous schedule for the current system.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.11 affineDepConsts

affineDepConsts[dp, v, dur, taus, alphas, opts] computes the constraints due to dependency do, on variable v, with duration dur, coefficients taus and alpha, and options opts.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.12 checkSchedOptions

checkSchedOptions[options] checks the options against the options of scd
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.13 constOf

constOf[symbol] gives the value of symbol
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.14 depComponents

depComponents[] computes the strongly connected components of the dependence graph
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.15 durationsNonZero

durationsNonZero is an option of scd. It gives the rank in the dependence table of the dependences the duration of which is non zero. Those durations

are assumed to be 1. A negative rank is counted from backward from the end of the dependence table.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.16 depCycles

depCycles[] computes the elementary cycles in the dependence graph. depCycles[selfDep→False] computes the elementary cycles, without the self dependences
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.17 depGraph

depGraph[] computes and displays the dependence graph of $result in the format of the Combinatorica package. This function assumes that the dependence graph has already been computed. See also Options[depGraph]. Inputs are green, outputs are blue and selfvertices are red.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.18 depGraphViz

depGraphViz[] computes and displays the dependence graph of $result in the format of the GraphViz package. This function assumes that the dependence graph has already been computed. See also Options[depGraph]. Inputs are green, outputs are blue and selfvertices are red. Option displaySchedule allows the schedule to be displayed.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.19 displaySchedule

displaySchedule is an option of depGraphViz that adds the schedule of all variables
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.20 showViz

showViz[] draws a dot file for the alpha program
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.21 variables

variables is an option of several functions, for example depGraph or appSched. Default value is all, meaning all variables of a system
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.22 extraEdges

Option of depGraph. Seems to be unused...
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.23 variables

Option of depGraph. Default value is all. Allows one to speficy the list of variables to which the graph is restricted.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.24 factor

Option of depGraph. Default value is 1. Gives a size factor for arrow heads and points of the dependence graph.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.25 labelSize

Option of depGraph. Default is 0.1. Size of the labels.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.26 labelFactor

Option of depGraph. Default is 1.2. Gives the distances of the labels from the center of the graph.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.27 DomDimension

DomDimension[d] gives the dimension of domain d, including the parameter dimension (see DomTrueDimension to get the dimension with de parameter dimension)
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.28 DomSingletonQ

DomSingletonQ[sys, d] is True if the true dimension of d is 0 and if d is the universal domain. DomSingletonQ[d] is equivalent to DomSingletonQ[$result,d]
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.29 DomTrueDimension

DomTrueDimension[sys,d] gives the dimension of domain d less the dimension of the parameter space of sys. DomTrueDimension[d] is equivalent to DomDimension[$result,d]
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.30 eliminateVar

eliminateVar[v] removes variable v, after checking that the variable is useless
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.31 encodeSchedule

encodeSchedule[sys, params, sol, opts] returns the encoding of the schedule sol of system sys, with parameters params, using options opts. Among the options of encodeSchedule, one option allows one to choose between the

Farkas format and the Vertex format
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.32 equations

equations is an option value of the option variables of depGraph. If True, the variables of are ordered in the order given by the equations, with input first.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.33 ishow

ishow[] pretty-prints $result, without calling the external C program. It supports some features of the imperative form (such as aliases) which ashow does not support.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.34 isIdentityOnDim

isIdentityOnDim[mat,{1,3}] (for instance) checks that matrix mat1 is identity on dimension 1 and 3
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.35 getLinPart

getLinPart[m] computes the mma form of the linear part of the Alpha matrix m, considered as an affine function. Note: use getLinearPart instead.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.36 getUseSchedule

getUseSchedule[x] obtains the schedule of the use entry of a dependency table x, by looking in the global variable $scheduleLibrary.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.37 loadScheduleLibrary

loadScheduleLibrary[sys] loads in global variable $scheduleLibrary the content of the file sys.scdlib
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.38 matrixTransPart

matrixTransPart[m] computes the mma form of the translation part of the Alpha matrix m, considered as an affine function.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.39 parameterRules

parameterRules is an option of loadScheduleLibrary. Default value is {}. It contains replacement rules for the parameters of the schedule which is loaded. For example, if option parameterRules → {"K"→2} is set, then parameter "K" will be replaced by value 2 while reading the schedule.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.40 periodicFactor

periodicFactor is an option of getUseSchedule. Its default value is 1.The first parameter of the schedule of a subsystem is multiplied byfactor
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.41 saveScheduleLibrary

saveScheduleLibrary[sys] saves the content of global variable $scheduleLibrary in the file sys.scdlib. saveScheduleLibrary[] saves the content of $scheduleLibrary in the file name.scdlib, where name is the name of the system currently contained in $result. To save only the schedule of the system in $result, use saveScheduleLibrary[onlyMainSystem → True].
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.42 onlyMainSystem

onlyMainSystem is an option of saveScheduleLibrary. If True, only the schedule of the system in $result is saved, otherwise (default) all schedules in $scheduleLibrary are saved.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.43 timeMinSchedConstraints

Timeminschedconstraints[] computes the constraints needed to minimize the total execution time of $result. timeMinSchedConstraints[var_String] computes only the dependence constraints for variable var. Options of timeMinSchedConstraints are sameLinearPart (default False), addConstraints ({}) and verbose (False).
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.44 saveSchedule

saveSchedule is an option of scd. If True, $schedule is saved in file sys.scd, where sys is the name of the scheduled system.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.45 scd

***scd[] finds out the schedule that minimizes the total computation time for a bounded Alpha program. Fails if the system is not bounded. scd can also be used to find a multidimensional schedule, or a dataflow schedule. An option allows a structured schedule to be found.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.46 periods

periods is a parameter of scd. It is the list of periods associated with the subsystems, whenever these subsystems contain the parameters ”$P”.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.47 onlyUseDep

onlyUseDep is an option of scd that specifies the use dependencies that are to be kept. Default value is all, otherwise, it is a list of integers.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.48 simplex

simplex[o,c,v] finds the minimum of function o on contraints c (list) for list of variables v. Variables are not assumed to be linear
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.49 slackOf

slackOf[listOfConstraints] gives the difference between the lhs and the rhs in the constraints, using the solution contained in $scheduleMDSol[1]. This function allows somehow to explore a given schedule.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.50 slg

slg[g] shows graph g. slg[g,Directed] shows g as a directed graph. This function is a slight modification of ShowLabeledGraph.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.51 pal

Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.52 sortEquations

sortEquations[sys:_system] returns a the same system where the equation have been reordered in such a way that all computation executed simultaneously (with duration 0) can be executed in the textual order, it currently assumes that the time is one dimensionnal and the first indice.

Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.53 zpolDomainQ

zpolDomainQ[d] is true if d is a domain with z-polyhedra
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.54 matrix2mma

matrix2mma[m] computes the mma form of the Alpha matrix m.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.55 eqsDomain

Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.56 statScheduleConstraints

statScheduleConstraints[] does as scheduleConstraints, but provides a few statistics
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.57 summaryScheduleConstraints

summaryScheduleConstraints[] computes the constraints needed to minimize the total time, and projects these constraints on the Input and Output variables.
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.3.58 variablesOf

variablesOf[exp] gives the list of variables in the list of expression exp
Defined in file: Alpha/VertexSchedule.m
Package: Alpha‘VertexSchedule‘

8.4 The Alpha‘FarkasSchedule‘ package

8.4.1 FarkasSchedule

Package. Schedule for alpha program with Farkas method, These function should be used through the schedule[] interface (Package Schedule.m)
Defined in file: Alpha/FarkasSchedule.m
Package: Alpha‘FarkasSchedule‘

8.4.2 farkasSchedule

(+) farkasSchedule[sys_Alpha‘system], finds the schedule of alpha system <sys>. farkasSchedule[] finds the schedule of $result and puts the result in $schedule. farkasSchedule[sys_Alpha‘system, {options}] calls farkasSchedule with non default options. Options[farkasSchedule] provides the options of schedule, and ?option provides information on option. ?$schedule provides info on output formats of farkasSchedule). By default, the schedule is affine by variable The schedule computation may take a long time (2 minutes for 10 instructions). More information is available in the file $MMALPHA/doc/user/Scheduler_user_manual.ps.
Defined in file: Alpha/FarkasSchedule.m
Package: Alpha‘FarkasSchedule‘

8.4.3 multiSched

(+) multiSched[sys_Alpha‘system], finds a multiDimensionnal schedule alpha system <sys> using the farkasSchedule scheduling function at each dimension. multiSched[] finds the schedule of $result and puts the result in $schedule. multiSched[sys_Alpha‘system, {options}] calls schedule with non default options. Options[multiSched] provides the options of schedule, and ?option provides information on option. See also: applySchedule.
Defined in file: Alpha/FarkasSchedule.m
Package: Alpha‘FarkasSchedule‘

Chapter 9

Uniformization

9.1 The Alpha‘Pipeline‘ package

9.1.1 Pipeline

Package. Contains the definition of the pipeline transformation. Containes the funcitons pipeline, pipeall, pipeIO
Defined in file: Alpha/Pipeline.m
Package: Alpha‘Pipeline‘

9.1.2 pipeline

pipeline[position,pipespec] pipelines in system $result expression given by position according to pipeline specification pipespec, and puts the result in $result. pipeline[sys, position,pipespec] does the same to program contained in symbol sys. Parameter pipespec provides the name of the new variable, and the pipeline direction in the textual form "newname.translation" (example: "B.(i,j→i,j+1)" means that new pipeline variable is B and data movement is (0,1)). Parameter position gives the position of the expression in the program, using the conventions of the Mathematica function Position (see getPart).
Defined in file: Alpha/Pipeline.m
Package: Alpha‘Pipeline‘

9.1.3 pipeall

pipeall[var, exp, pipespec] pipelines in program $result all occurrences of expression exp within equation whose left-hand side is var according to

pipeline specification pipespec, and returns the result in $result. pipeall[sys, var, exp, pipespec] does the same to program contained in symbol sys. pipeall[var, exp, pipespec, bd] pipelines exp of var definition using pipespec, but extend the domain of pipeline to the boundary given by domain bd. Parameter var can be either a single variable, a list of variables, or the empty list, in which case the expression is pipelined in all the program simultaneously. The parameters can be specified either in textual form or as AST. For example, pipeall["X","a.(i,j,k→i,j+1,k)","A.(i,j,k→i,j+1,k)"] pipes all occurences of "a.(i,j,k→i,j+1,k)" in the definition of X, using the pipeline specification "A.(i,j,k→i,j+1,k)". The pipeline specification contains the name to be used for the pipeline variable (here "A"), and the direction of pipeline given as a translation (here, vector (0,1,0)). Similarly, pipeall["X","a.(i,j,k→i,j+1,k)","A.(i,j,k→i,j+1,k)", "{i,j,k|i>=0}"] pipes all occurences of "a.(i,j,k→i,j+1,k)" in the definition of X, using the pipeline specification "A.(i,j,k→i,j+1,k)", withing the boundary "{i,j,k|i>=0}"]. In case of error, pipeall returns Null, and leaves $result unchanged. For programming purposes, pipeall may also be called with the form pipeall[sys,var,expression,pipespec], where expression and pipeline are abstract syntax trees. Parameter expression can also be replaced by the occurrence number of the expression. See also the function pipeInfo which gives hints about which pipeline commands to perform.
Defined in file: Alpha/Pipeline.m
Package: Alpha‘Pipeline‘

9.1.4 pipeAll

pipeall[var, exp, pipespec] pipelines in program $result all occurrences of expression exp within equation whose left-hand side is var according to pipeline specification pipespec, and returns the result in $result. pipeall[sys, var, exp, pipespec] does the same to program contained in symbol sys. pipeall[var, exp, pipespec, bd] pipelines exp of var definition using pipespec, but extend the domain of pipeline to the boundary given by domain bd. Parameter var can be either a single variable, a list of variables, or the empty list, in which case the expression is pipelined in all the program simultaneously. The parameters can be specified either in textual form or as AST. For example, pipeall["X","a.(i,j,k→i,j+1,k)","A.(i,j,k→i,j+1,k)"] pipes all occurences of "a.(i,j,k→i,j+1,k)" in the definition of X, using the pipeline specification "A.(i,j,k→i,j+1,k)". The pipeline specification contains the name to be used for the pipeline variable (here "A"), and the direction of pipeline given as a translation (here, vector (0,1,0)). Similarly,

pipeall["X","a.(i,j,k→i,j+1,k)","A.(i,j,k→i,j+1,k)", "{i,j,k|i>=0}"] pipes all occurences of "a.(i,j,k→i,j+1,k)" in the definition of X, using the pipeline specification "A.(i,j,k→i,j+1,k)", withing the boundary "{i,j,k|i>=0}"]. In case of error, pipeall returns Null, and leaves $result unchanged. For programming purposes, pipeall may also be called with the form pipeall[sys,var,expression,pipespec], where expression and pipeline are abstract syntax trees. Parameter expression can also be replaced by the occurrence number of the expression. See also the function pipeInfo which gives hints about which pipeline commands to perform.
Defined in file: Alpha/Pipeline.m
Package: Alpha‘Pipeline‘

9.1.5 pipeIO

pipeIO[var,exp,pipespec,boundary] pipelines in $result I/O expression exp occuring in variable var to plane boundary according to pipeline specification pipespec. pipeIO[sys,var,exp,pipespec,boundary] does the same to program contained in symbol sys. Parameters var and exp are strings. Parameter pipespec is the pipeline specification string in the form "newVar.direction" (example: "X1.(t→t-1)"). Parameter boundary is given in textual form (example: "
{i,j | 2i+3j+4>=0
}"). For example, pipeIO["y","x","X.(t,n→t+1,n+1)","
{t,n|n>=-1
}"] pipes all occurences of x in definition of y in $result to plane "
{t,n | n>=-1
}", with new name X, and pipeline direction (1,1). If something wrong happens, pipeIO returns the system unmodified. For programming purpose, exp, pipespec and boundary can be given as AST instead of textual form. Also, the position of the expressions to pipeline may be given. See test file pipeIOT.m in $MMALPHA/test.
Defined in file: Alpha/Pipeline.m
Package: Alpha‘Pipeline‘

9.2 The Alpha‘Control‘ package

9.2.1 Control

Package. Contains the definition of functions related to control signal generation: temporalCaseQ[], spatialCaseQ[], spaceTimeCase[], spaceTimeDe-

composition[], needsMuxQ[], makeMuxControl[], makeAllMuxControl[], isControlEquQ[], controlVars[].
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.2 makeOneMuxControl

makeOneMuxControl[sys, tpos, spos, var, options] generates a multiplexer for the definition of var.
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.3 makeAllMuxControl

makeAllMuxControl[sys, tpos, spos, options] generates multiplexers and their control variables for all the variables in the system sys needing it. The system sys must be in space/time form (see spaceTimeDecomposition). The modified system is returned. makeAllMuxControl[tpos, spos, options], applies to $result and modifies it.
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.4 temporalCaseQ

temporalCaseQ[sys, expr, post, poss] checks whether or not the case expression expr of sys is a temporal case (all alternatives are defined over the same spatial domain.) post is the list of positions of temporal indices and poss is the list of positions of spatial indices. WARNING: This function checks pure temporal case (no condition involving space can change in the branches of the case)
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.5 spatialCaseQ

spatialCaseQ[sys, expr, post, poss] checks whether or not the case expression expr of sys is a spatial case (i.e., can be rewritten using conditions on spatial indices only). post is the list of positions of temporal indices, and poss is the list of positions of spatial indices.
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.6 spaceTimeCase

spaceTimeCase[sys, varname, tpos, spos] unrolls the normalized case-based definition of local variable varname to a double case spatial then temporal. spaceTimeCase[varname, tpos, spos] applies to $result and modifies it.
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.7 spaceTimeDecomposition

spaceTimeDecomposition[sys, tpos, spos] transforms all the local variables of sys into their space-time case form. spaceTimeDecomposition[varname, tpos, spos] applies to $result and modifies it.
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.8 needsMuxQ

needsMuxQ[exp] returns True if expr of $result needs a multiplexer, False otherwise.
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.9 makeMuxControl

makeMuxControl[var, ctrlVar, tpos, spos] builds the mux control variables for variable var. It is restricted to cases with two branches. WARNING: DOES not check that the temporal cases are binary.
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.10 isControlEquQ

isControlEquQ[equation] checks that equation is a controler equation.
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.11 controlVars

controlVars[sys] returns the list of control variables in system sys. controlVars[] applies to $result. A control variable is defined by a space/time case with two time branches equal to True and False.

Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.2.12 makeBinaryCases

makeBinaryCases[sys, var, opts] replaces in the definition of var, all case statements with more than 2 branches by binary cases. makeBinaryCases[var, opts] does the same to $result.
Defined in file: Alpha/Control.m
Package: Alpha‘Control‘

9.3 The Alpha‘PipeControl‘ package

9.3.1 eq2ge

Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.2 eq2le

Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.3 PipeControl

This package contains functions to transform case expressions for the pipeline of control variables. These functions are: pipeControl, pipeAllControl, report, uniformizeMatrix, uniformizeDep, findSepHalfSpace, findPipeControl, domExplode.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.4 pipeAllControl

pipeAllControl[sys, options] generates pipeline control signals for all the control variables present in sys (see pipeControl[]) and returns the modified system. pipeAllControl[options] applies to $result.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.5 pipeControl

pipeControl[var] generates a pipelined control signal from the definition of variable var. pipeControl[sys, var] does the same to program contained in symbol sys. pipeControl checks that var is defined using a doubly nested case expression. The current version only covers the case when var is defined on a 2-dimensional domain, and we assume that the first dimension is the time. The new variable is named varPipe. pipeControl may not succeed for the following reasons: the pipeline direction is searched among the halfspaces found in the separating domains of the inside case expression and this heuristics may fail. In practice, we believe that it is reasonable...
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.6 pipeInfo

pipeInfo[occ:{__Integer}] returns information about pipelining occurrence occ in $result. If the occurrence can be piped, then pipeInfo returns a list formed by the occurence and the pipevector. If not, it returns $Failed. pipeInfo[] applies pipeInfo on all dependencies of $result. pipeInfo[sys, occ], or pipeInfo[sys] do the same to program sys. pipeInfo has an option execute → True|False, When this option is set, pipeInfo executes the first pipe it has found
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.7 execute

Option of pipeInfo. If True, pipeInfo tries to pipeline system, according to the informations it has found
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.8 pipeConstants

Option of pipeInfo. If True, pipeInfo tries to pipeline constants
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.9 pipeVars

pipeVars[] executes pipeInfo until it converges, a bounded number of times
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.10 iterations

Option of pipeVars. Gives the maximum number of times that pipeInfo can be called.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.11 uniformizeMatrix

uniformizeMatrix[mat, contextDom] tries to find out if mat can be made uniform in the context defined by contextDom, and returns the uniform matrix found.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.12 constantizeMatrix

constantizeMatrix[mat, contextDom] eliminates as many indexes as possible in the context defined by contextDom, and returns the matrix found.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.13 constantizeOccur

constantizeOccur[occ] tries to transform into constants the dependency occ in program $result, by eliminating as many redundant indexes as possible. The parameter occ specifies the position of the dependency, and can be obtained using the getOccurs function.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.14 mkUniform

mkUniform[occ] uniformizes the dependence occ in $result. This function should be called only after getting the result of uniformizeOccur

Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.15 uniformizeOccur

uniformizeOccur[occ] tries to uniformize the dependency occ in program $result. The parameter occ specifies the position of the dependency, and can be obtained using the getOccurs function.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.16 reportDep

reportDep[repFile, occ, opts] analyzes an dependency given by its occurrence occ, with options opts, and writes the report in the output stream repFile. This function is called by report.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.17 report

report[] analyzes all dependences of $result and gives information on it in file current.report. report has options. report[name] analyzes a program in file name.alpha and puts the result in file name.report.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.18 findSepHalfSpace

findSepHalfSpace[exp] finds a list of separating hyperplanes for binary case expression exp. The method is heuristic in that the candidate hyperplanes are taken within the constraints defining the case expression. exp can be either a String or an Ast
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.19 findPipeControl

findPipeControl[x] finds if symbol x is defined using a doubly nested case expression, and returns the list of separating hyperplanes with time coefficient equal to 1, of the second level case expression. This function is used

in the control signal generation.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.20 domExplode

domExplode[dom] produces a list of halfspace from a domain definition.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.21 domExplode1

domExplode1[dom] produces a list of halfspace from a domain definition.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.22 route

route[occ, {var, v}, dom1, dom2] routes occurence occ in $result, between dom1 and dom2, with routing vector v.
Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

9.3.23 delocalizeControl

Defined in file: Alpha/PipeControl.m
Package: Alpha‘PipeControl‘

Chapter 10

Back end process

10.1 The Alpha‘ToAlpha0v2‘ package

10.1.1 ToAlpha0v2

ToAlpha0v2: Package. Contains functions for the transformation of scheduled programs into Alpha0v2 form: decomposeSTdeps, makeInputMirrorEqus[], toAlpha0v2[]
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.2 needSeparation

debugUse
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.3 toAlpha0v2

toAlpha0v2[sys_Alpha‘system,options_List] converts a scheduled program ‘sys’ (see schedule[] and applySchedule[]) to Alpha0v2 form. The program ‘sys’ is assumed to be such that all the local variables have the same dimension (see uniformizeDims[]) and all the dependencies are uniform. This function calls in sequence spaceTimeDecomposition[] makeAllMuxControl[], pipeAllControl[] and decomposeSTdeps[]. See these transformations for further information. WARNING: if there is more than one space index then pipeAllControl[] doesn’t work yet. If sys is not present then the transformations apply to $result and modify it. The option list is optional.

Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.4 steps

steps is an option of toAlpha0v2. It gives a list of numbers, each one corresponding to a step.
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.5 makeSimpleExpr

makeSimpleExpr[sys] renames all the subexpressions which are not simple until the program as no more composed expression (in the sense of the analysis performed by needSeparation). Currently, the expression simplified are only the one involving multiplexer and operators
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.6 decomposeSTdeps

decomposeSTdeps[sys] splits, in sys, each dependency involving space and time into two dependencies, one on space, the other on time. decomposeSTdeps[] applies to $result and modifies it.
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.7 makeInputMirrorEqus

makeInputMirrorEqus[sys_Alpha‘system,options_List] adds to system ‘sys’ mirror equations fot the inputs. This low-level transformation is needed before the translator to AlpHard is invoked. ‘sys’ is optional, defaults to $result, and the option list is optional, too (see Options[makeInputMirrorEqus]).
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.8 reuseCommonExpr

reuseCommonExpr[sys_Alpha‘system] attemps to gather expressions used several time by adding local variables for these expressions in system ‘sys’.

reuseCommonExpr[sys_Alpha‘system,var_String,expr_] specifically add variable ‘var’ for expr ‘expr’ (which s in ast form). Currently only RHS of equation are scanned up for multiple uses
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.9 integerToBooleanSyst

integerToBooleanSyst[<syst_Alpha‘system>] Function, change all * in and and all + in or. Returns the resulting AST
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.10 booleanToIntegerSyst

booleanToInteger[<syst_Alpha‘system>] Function, change all and in * and all or in +. Returns the resulting AST
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.11 correctIdEqs

debug
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.1.12 splitMax

splitMax[sys]returns the system sys in which the Max4 has been replaced by calls to Max operators, splitMax[] applies this to $result
Defined in file: Alpha/ToAlpha0v2.m
Package: Alpha‘ToAlpha0v2‘

10.2 The Alpha‘Alphard‘ package

10.2.1 alpha0ToAlphard

alpha0ToAlphard[sys,controlSigList] translate the Alpha0 system ‘sys’ into an Alphard library, ‘controlSigList’ is the list of names of the control signals. alpha0ToAlphard[controlSigList] happens on $result and modify $result and $library

Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.2 alpha0ToAlphardModule

alpha0ToAlphardModule[sys,listRegion,controlSigList] translate the Alpha0 Module ‘sys’ into an Alphard library. ‘listRegion’ is the structural information resulting from the function getArrayDomains[sys], ‘controlSigList’ is the list of names of the control signals.
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.3 alphardFirstStep

alphardFirstStep[] returns an integer which is the first logical steps of the alphard architecture. The reset signal should be sent at this step.
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.4 alphardTimeLife

alphardTimeLife[sys:_Alpha‘system] returns the time domain spanned by the hardware. This function should be called on the module of an Alphard program after the parameter have been assigned (your can call it before the assignement, but the result will be more difficult to interpret), alphardTimeLife[] applies on $library (except the last program) and take the union
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.5 getArrayDomains

getArrayDomains[sys] computes all the spatial region with a different behaviour determined by the system sys (on each spatial region, cells are idential). The system sys must be in Alpha0 form. getArrayDomains[] applies to $result.
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.6 buildControler

buildControler[sys,signalList] builds the subsystem which initializes all the control signals of 'sys' indicated by 'signalList'. The system 'sys' must be in Alpha0 form. buildControler[signalList] takes $result as default value for 'sys'
Defined in file: Alpha/Alphard.m
Package: Alpha`Alphard`

10.2.7 isSpaceDepQ

isSpaceDepQ[dep] checks whether the Alpha dependance 'dep' is a space dependance (in Alpha0). 'dep' is an AST
Defined in file: Alpha/Alphard.m
Package: Alpha`Alphard`

10.2.8 isConnexionEqQ

isConnexionEqQ[eq] checks whether the Alpha equation 'eq' is a connexion equation (in Alpha0). 'eq' is an AST
Defined in file: Alpha/Alphard.m
Package: Alpha`Alphard`

10.2.9 buildOneCell

buildOneCell[sys,dom1,posList] return the Alphard cell (extracted form system 'sys'), occuring on the spatial region indicated by 'dom1' and which computes expressions indicated by the list of poisiton 'posList' in AST 'sys'. 'dom' and 'posList' can be obtained by the function getArrayDomains[sys]. buildOneCell[sys,dom1,posList,controlSignals] do the same with the additionnal information of the list of control signals in 'controlSignals'.
Defined in file: Alpha/Alphard.m
Package: Alpha`Alphard`

10.2.10 isMirrorEqQ

isMirrorEqQ[eq] checks whether the Alpha equation 'eq' is a Mirror equation (in Alpha0). 'eq' is an AST
Defined in file: Alpha/Alphard.m
Package: Alpha`Alphard`

10.2.11 buildInterface

buildInterface[sys] returns two Alphard systems: the interface and the module which correspond to system sys. The interface has the same inputs and outputs as sys and simply calls the module with scheduled input and output (Mirror variables for i/o). buildInterface[] applies on $result and modifies $result and $library.
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.12 isModuleQ

isModuleQ[sys] checks whether the Alpha system ‘sys’ is a module or not
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.13 removeSystem

removeSystem[id_String] Remove the system named ‘id’ from the library $library (changes $program and $result if system ‘id’ is also $result).
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.14 simplifyConnexions

simplifyConnexions[sys] translate connexion syntax of connexion from Alpha0 mode to Alphard mode in system ‘sys’, return the modified system. simplifyConnexions[] simplify connexion of $result
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.15 normalizeIdDep

normalizeIdDep[sys], normalize identity dependencies: replace in system ‘sys’ all occurences of Alpha‘var[a___] which is not surrounded by an Alpha‘affine by Alpha‘affine[Alpha‘var,identity] and return the modified system. normalizeIdDep[] apply the function to $result and assigns the result to $result. e.g A[t,p]=B is replaced by A[t,p]=B[t,p]. Warning, the transformation is not applied to the input arguments of Alpha‘use
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.16 normalizeIdDepInEq

normalizeIdDep[sys,eq] normalize identity dependencies: replace in equation 'eq' of system 'sys' all occurences of Alpha‘var[a___] which is not surrounded by an Alpha‘affine by Alpha‘affine[Alpha‘var,identity] and return the modified equation
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.17 normalizeIdDepLib

normalizeIdDepLib[] applies normalizeIdDep[] to all subsystems of the library, assigns it to $library and return the resulting Library (change also $result)
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.18 isolateOutputList

isolates outputs of a system in the library, so that an output is never used inside its generating system (necessary for VHDL)
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.19 isolateOutput

isolates outputs of a system in the library, so that an output is never used inside its generating system (necessary for VHDL)
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.20 structureFrom

structureFrom[indices_List[__String],listAllVars_List[__String]] build a subsystem from $result which defines all the variables in 'listAllVars', and for which the indices in 'indices' will be the extension indices of the use set in $result. All variables must give the same domain when projecting then on the indices 'indices'. Can also be called in the forllowing form : structureFrom[indices], structureFrom[sys,indices], structureFrom[sys,indices,listStructVar], structureFrom[sys,indices,listStructVar,newName]. Warning, not tested
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.21 setOutputVar

setOutputVar[sys1_systel,var1_String] returns sys1 in which var1 in moved from local vars to output vars (added as last output var). setOutputVar[var1] operates on $result. WARNING this function changes the semantics of the program (the use calling this program are not valid anymore)
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.2.22 insertFunction

inserFunction[var1_String, func1_String] insert the function func1 in the definition of var var1: if the definition of var1 was var1=expr1, the new definition is var1=func1(var1). This is mainly used for the introduction of functions like truncateLSB for bit level description of the operations.
Defined in file: Alpha/Alphard.m
Package: Alpha‘Alphard‘

10.3 The Alpha‘Vhdl2‘ package

10.3.1 Vhdl2

Alpha‘Vhdl2‘ contains the Vhdl generator. The main function is a2v.
Defined in file: Alpha/Vhdl2.m
Package: Alpha‘Vhdl2‘

10.3.2 $vhdlCurrent

vhdlCurrent is a global variable owned by Vhdl, which contains the current program
Defined in file: Alpha/Vhdl2.m
Package: Alpha‘Vhdl2‘

10.3.3 $vhdlOutputFile

Defined in file: Alpha/Vhdl2.m
Package: Alpha‘Vhdl2‘

10.3.4 a2v

a2v[] translates in Vhdl all the files contained in $library. a2v[lib] translates in Vhdl all the files contained in library lib. a2v[sys] translates in Vhdl the system sys. a2v has many options (see Options[a2v]). a2v[elem,tinit] runs a2v on library or system elem, and fixes the initial value of the time to the integer value tinit.
Defined in file: Alpha/Vhdl2.m
Package: Alpha‘Vhdl2‘

10.3.5 bitWidth

bitWidth[sys,var] gives the bit width of variable var. Default value of sys is $result.
Defined in file: Alpha/Vhdl2.m
Package: Alpha‘Vhdl2‘

10.3.6 bitWidthOfExpr

bitWidth[sys,expr] gives the bit width of an expression. Default of sys is $result.
Defined in file: Alpha/Vhdl2.m
Package: Alpha‘Vhdl2‘

10.3.7 getVhdlType

getVhdlType[sys,var] return the vhdl type of the element of array var
Defined in file: Alpha/Vhdl2.m
Package: Alpha‘Vhdl2‘

10.3.8 showVhdl

showVhdl[] prints the Vhdl generated for the system in $result. showVhdl[s] show the content of the file s.vhd.
Defined in file: Alpha/Vhdl2.m
Package: Alpha‘Vhdl2‘

10.3.9 stim

stim[] post processes all the simuli files (replace blanks by zeros). These files are generated from a C code in hexadecimal and there is no format in C to write zeros in for the first digits.

Defined in file: Alpha/Vhdl2.m
Package: Alpha‘Vhdl2‘

10.4 The Alpha‘VhdlTestBench‘ package

10.4.1 VhdlTestBanch

Package,contains definition of funtion for generating test benches for VHDL generated from Alphard
Defined in file: Alpha/VhdlTestBench.m
Package: Alpha‘VhdlTestBench‘

10.4.2 vhdlTestBenchGen

vhdlTestBenchGen[sys] generates a VHDL test bench for the VHDL component generated from sys with the a2v command. sys mustbe a valid Alphard system, it works if sys is a cell, a module or a controller (not an interface), as for a2v, the parameters must be assigned. vhdlTestBenchGen[] applies on $result
Defined in file: Alpha/VhdlTestBench.m
Package: Alpha‘VhdlTestBench‘

10.4.3 mkVhdlLoop

test
Defined in file: Alpha/VhdlTestBench.m
Package: Alpha‘VhdlTestBench‘

10.4.4 mkVhdlLowerBound

test
Defined in file: Alpha/VhdlTestBench.m
Package: Alpha‘VhdlTestBench‘

10.4.5 mkVhdlUpperBound

test
Defined in file: Alpha/VhdlTestBench.m
Package: Alpha‘VhdlTestBench‘

Chapter 11

Utilities

11.1 The Alpha‘MakeDoc‘ package

11.1.1 Note regarding the documentation of MMALPHA

The MMALPHA software is (partially) documented in various ways.[1]

- A `html` documentation is included in the distribution. It can be accessed from the `welcome.html` file in the `$MMalpha` directory.
- A *getting started* document is available in form of a `pdf` file. It is accessible from the initial `html` welcome page and is located in:

  `$MMAlpha/doc/QuickStart/AlphaStart.pdf`

  This document is the most up-to-date tutorial for MMALPHA. It does not describe all features of MMALPHA, but normally, allows a user to discover the main features of the software.
- A reference manual is being developped in:

  `$MMAlpha/doc/ReferenceManual`

  and it is described hereafter.
- Other documentation files exist, but they are not guaranteed to be up-to-date.

[1] This chapter has been proof-read by Patrice Quinton on January 8, 2007.

The reference manual of MMALPHA is obtained through the use of commands in the `Makedoc` package. It is produced by executing the program called `genReferenceManual.m` which is situated in the same directory.

This program contains a set of calls to the function `doDoc` of the `Makedoc` package (see `?doDoc`). Each one of these calls reads a package – for example, `Alpha/Makedoc.m` – whose name is relative to directory

```
$MMALPHA/lib/Packages
```

and translates it into a LaTeX file – for example, `Makedoc.tex`, – in target directory `$MMALPHA/doc/ReferenceManual`.

The reference manual itself is edited manually. It is in the file:

```
referenceManual.tex
```

of the same directory. It contains basically input statements for the documentation files created by `doDoc`. This function (which is documented hereafter), has several options: one can either generate a standalone LaTeX file, or an input file (this helps for editing the documentation), one can specify a list of functions that are only to be documented, one can specify where the file has to be put, etc.

The reference manual is based on the usage and note statements which are contained in a given package. In there exists a file named `incl-textFile` in the target directory, then an include statement for this file is added in the generated LaTeX file. For example, the documentation that you are reading is in the file `incl-Makedoc.text` file, in directory:

```
$MMALPHA/doc/referenceManual
```

Therefore, when the `Makedoc.m` file is processed by the corresponding statement in the `genReferenceManual.m` program, i.e.:

```
doDoc["Alpha/MakeDoc.m", "MakeDoc.tex",
  targetDir -> Environment["MMALPHA"] <> "/doc/ReferenceManual"];
```

an input statement is included in the `Makedoc.tex` file which is generated.

The `doDoc` function tries to modify the content of the usage and note statements so that special symbols are correctly output in LaTeX. These transformations are described in the variable `repRules` of the package, and may be modified. Index statements are generated for each usage statement.

11.1.2 MakeDoc

MakeDoc is a package which contains functions for the automatic creation of a reference Manual. MakeDoc contains functions doDoc and makeDoc.
Defined in file: Alpha/MakeDoc.m
Package: Alpha‘MakeDoc‘

Functions of MakeDoc read all usage and note statements of a package, and for each one, they produce a Latex macro entry alphausage or alphanote. Before creating an entry, substitutions are performed on the text, so that special words (such as $schedule) and special characters (such as _) are transformed. The substitution list is in the private variable repRules in the file MakeDoc.m.

11.1.3 doDoc

doDoc[package, texFile] generates in file texFile the documentation LaTex file associated to a Mathematica file package. doDoc extracts the information contained in the usage and note descriptions placed at the beginning of file package, and produces an output Latex file. doDoc[package] automatically writes in a file which has the name package where the .m suffix is replaced by .tex. doDoc[Package, functionList] generates the documentation only for the functions enumerated in functionList (as Strings). doDoc has options fullLatex and callFile which allow a complete latex file, resp. a calling Latex file to be produced. If there exists a file incl-textFile in the target directory, then a Latex input statement is added in the documentation package.
Defined in file: Alpha/MakeDoc.m
Package: Alpha‘MakeDoc‘

11.1.4 fullLatex

fullLatex is an option of doDoc and makeDoc. Default value is False. If True, the latex file contains the preamble and the
end{document} statement. Otherwise, it should be used in an input file
Defined in file: Alpha/MakeDoc.m
Package: Alpha‘MakeDoc‘

11.1.5 targetDir

option of doDoc. By default, it is "". Gives the name of a directory where the latex file and the calling latex file should be written.

Defined in file: Alpha/MakeDoc.m
Package: Alpha‘MakeDoc‘

11.1.6 callFile

callFile is an option of doDoc and makeDoc. Default valus is False. If True, a latex calling file is created.
Defined in file: Alpha/MakeDoc.m
Package: Alpha‘MakeDoc‘

11.1.7 sourceDir

option of makeDoc. Default value is Null. If Null, the source package are those of $MMALPHA. Their names are in the file $MMALPHA/sources/ MaleDoc/List_modules. Otherwise, these names are prefixed by the value of the sourceDir option.
Defined in file: Alpha/MakeDoc.m
Package: Alpha‘MakeDoc‘

11.1.8 makeDoc

This function does not work, currently. Use doDoc instead. makeDoc[] updates the Reference Manual of MMALPHA by running the file genReferenceManual.m in directory
$MMALPHA/doc/ReferenceManual/
This produces new versions of the latex files describing the packages. The Reference Manual can be produced using Latex on referenceManual.tex. See also doDoc.
Defined in file: Alpha/MakeDoc.m
Package: Alpha‘MakeDoc‘

Chapter 12

Add-ons

12.1 The Alpha‘INorm‘ package

12.1.1 iNorm

iNorm[s:_system, k:_Integer, options] computes the imperative form of system s. k is the dimension of time. As usual, iNorm[k, options] means $result = iNorm[$result, k, options]. See Options[iNorm] for options.
Defined in file: Alpha/INorm.m
Package: Alpha‘INorm‘

12.1.2 rnf

rnf[s:_system] returns the R normal form of s.
Defined in file: Alpha/INorm.m
Package: Alpha‘INorm‘

12.2 The Alpha‘Lexicographic‘ package

12.2.1 domLexGreater

domLexGreater[v:_Matrix] returns a domain D such that for all z in D, z>v, where > means lexicographically greater. V is a parametrized vertex. If v is a n*(p+1) matrix, D is a n+p domain, and the p last dimensions are parameter dimensions. Example: domLexGreater[{{1,0},{0,2}}] returns {a,b,c | c+1<=a} | {a,b,c | a=c; 3<=b}, where c is the parameter index. domLexGreater[v:_Vector] is a shortcut for 0-dimensional parameter space. (i.e. domLexGreater[{0,0,1}] means domLexGreater[{{0},{0},{1}}].

Defined in file: Alpha/Lexicographic.m
Package: Alpha‘Lexicographic‘

12.2.2 domLexLower

domLexLower[v:_Matrix] returns a domain D such that for all z in D, z<v, where < means lexicographically lower. V is a parametrized vertex. If v is a n*(p+1) matrix, D is a n+p domain, and the p last dimensions are parameter dimensions. Example: domLexLower[{{1,0},{0,2}}] returns {a,b,c | a<=c-1} | {a,b,c | a=c; b<=1}, where c is the parameter index. domLexLower[v:_Vector] is a shortcut for 0-dimensional parameter space. (i.e. domLexLower[{0,0,1}] means domLexGreater[{{0},{0},{1}}].
Defined in file: Alpha/Lexicographic.m
Package: Alpha‘Lexicographic‘

12.2.3 lexGreaterQ

lexGreaterQ[a:_Matrix, b:_Matrix, p:_domain] returns True if a is lexicographicallygreater than b for any value of parameters in p. a and b are parametrized vectors.
Defined in file: Alpha/Lexicographic.m
Package: Alpha‘Lexicographic‘

12.2.4 lexGreaterOrEqualQ

lexGreaterOrEqualQ[a:_Matrix, b:_Matrix, p:_domain] returns True if a is lexicographically greater or equal than b for any value of parameters in p. a and b are parametrized vectors.
Defined in file: Alpha/Lexicographic.m
Package: Alpha‘Lexicographic‘

12.2.5 lexLowerQ

lexLowerQ[a:_Matrix, b:_Matrix, p:_domain] returns True if a is lexicographically lower than b for any value of parameters in p. a and b are parametrized vectors.
Defined in file: Alpha/Lexicographic.m
Package: Alpha‘Lexicographic‘

12.2.6 lexLowerOrEqualQ

lexLowerOrEqualQ[a:_Matrix, b:_Matrix, p:_domain] returns True if a is lexicographically lower or equal than b for any value of parameters in p. a and b are parametrized vectors.
Defined in file: Alpha/Lexicographic.m
Package: Alpha‘Lexicographic‘

12.3 The Alpha‘Matrix‘ package

Documentation revised on March 11, 2008

12.3.1 Matrix

The Alpha‘Matrix‘ package contains a few functions related to Alpha matrices. These functions are alphaToMmaMatrix, composeAffines, convHull, convexize, convexizeAll, deleteColumn, deleteRow, determinant, dropColumns, dropParameters, dropRows, emptyLinearPartQ, getLinearPart, getTranslationVector, hermite, hermiteL, hermiteR, inverseMatrix, identityQ, idLinearPartQ, idMatrix, inverseInContext, nullSpaceVectors, mmaToAlphaMatrix, nullLinearPartQ, simplifyAffines, solveDiophantine, smithNormalForm, squareMatrixQ, suppressRowNum, translationMatrix, translationQ, unimodularQ, composeAffines, inverseMatrix, unimodularQ, unimodularCompletion, unimodCompl.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.2 addLinSpace

Obsolete function, replaced by the generalized ChangeOfBasis.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.3 alphaToMmaMatrix

alphaToMmaMatrix[mat] translates a linear function from Alpha format into Mathematica format (list of list). If the linear function mat was affine (non nul constant part) this constant part is lost.
Example:
alphaToMmaMatrix[

matrix[3, 3, {i, j}, {{1, 2, 0}, {0, 3, 0}, {0, 0, 1}}]
]
gives {{1, 2}, {0, 3}}.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.4 canonicalProjection

Obsolete. canonicalProjection[dom, pos] returns a projected domain obtained by supressing the indices given by the integer list pos.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

OBSOLETE. Replaced by (generalized) ChangeOfBasis.

12.3.5 composeAffines

composeAffines[m1,m2] returns the composition of two affine matrices m1 and m2. These matrices are described in the Alpha format.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.6 convHull

convHull[ldom] returns the convex hull of a (possibly empty) list of domains ldom.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

If the input list is empty, returns an empty list.

12.3.7 convexize

convexize[domain] returns the convex hull of domain if domain is convex, otherwise returns domain unmodified. Used to convexize a union of domains (try to use DomConvex instead).
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.8 convexizeAll

convexizeAll[sys] tries to convexize all the non-convex domains of sys (default $result) and returns the modified system.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.9 deleteColumn

deleteColumn[m,k] removes the k-th column of the Alpha matrix m. It removes also index k of m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.10 deleteRow

deleteRow[m,k] removes the k-th row of the Alpha matrix m. It does not touch the index list of m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.11 determinant

determinant[m] gives the determinant of a square Alpha matrix. The result can be integer or rationnal, the constant part is not taken into account.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.12 dropColumns

dropColumns[m,n] removes n columns of the Alpha matrix m, from front if n is positive, from end if n is negative. This function does not touch the dimensions nor the indexes of the matrix.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.13 dropParameters

dropParameters[sys,m] removes the rows of Alpha matrix m corresponding to parameters of system sys (default $result). If there are too many parameters, dropParameters returns m unchanged.

Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.14 dropRows

dropRows[m,n] removes n rows of the Alpha matrix m, from front if n is positive, from the end otherwise. This function does not touch the dimensions nor the indexes of the matrix.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.15 emptyLinearPartQ

emptyLinearPartQ[m] is True if the linear part of matrix m is empty, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.16 getLinearPart

getLinearPart[m] returns the linear part of an affine function m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.17 getTranslationVector

getTranslationVector[m] returns the translation vector of a square affine function m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.18 hermite

hermite[m] returns the left hermite decomposition {h,u} (as Alpha matrices) of m1 (such that m1=composeAffines[h,u]).
WARNING: Currently, this function only deals with linear function (the constant part is ignored).
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.19 hermiteL

hermiteL[m] performs the left Hermite decomposition of the Mathematica matrix m and returns mathematica matrices {H,Q} where Q unimodular, H upper triangular and m = Dot[H,Q].
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.20 hermiteR

hermiteR[m] computes the right Hermite decomposition of the Mathematica matrix m and returns Mathematica matrices {H,Q}, where Q is unimodular, H is lower triangular, and m=Dot[Q,H].
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.21 inverseMatrix

inverseMatrix[m] computes and returns the inverse of a square affine matrix m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.22 identityQ

identityQ[m] is True if an Alpha affine function m is the identity, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.23 idLinearPartQ

idLinearPartQ[m] is True if the linear part of the affine function given as Alpha matrix m is the identity, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.24 idMatrix

idMatrix[idx_List,idy_List] returns the transformation matrix for dependence (idx→idy), where idx and idy are lists of index names. It is assumed that names in idy are also in idx.

Example:
idMatrix[{"i","j"},{"i","i","j"}] =
matrix[4, 3, {"i","j"}, {{1,0, 0}, {1, 0, 0}, {0, 1, 0}, {0, 0, 1}}].
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.25 inverseInContext

inverseincontext[m,d] computes the inverse matrix of Alpha matrix m in context with domain d, i.e., taking into account the lineality space defined by d.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.26 isIdLinearPart

obsolete form of idLinearPartQ.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.27 isNullLinearPart

obsolete form of nullLinearPartQ.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.28 nullSpaceVectors

nullSpaceVectors[mat] gives the list of vectors of the null space of Alpha matrix mat.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.29 mmaToAlphaMatrix

mmaToAlphaMatrix[m] returns the Alpha form of a Mathematica matrix m. mmaToAlphaMatrix[m,i] returns the Alpha form of matrix m with index (list of strings) i. mmaToAlphaMatrix[m,c:List[___]] returns the Alpha form of a linear function Z → mZ+c.
Example:
mmaToAlphaMatrix[{{1, 2},{0, 3}},{1,4}]

gives the Alpha matrix which represents the affine function (i,j→i+2j+1,3j+4). mmaToAlphaMatrix[m,c,i] returns the Alpha form of linear function Z → mZ+c, with index (string list) i.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.30 nullLinearPartQ

nullLinearPartQ[m] is True if the linear part of the affine function given as Alpha matrix m is null, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.31 solveDiophantine

solveDiophantine[a,b] solves the linear diophantine system of equations aX = b where a is a MMA matrix, and b a MMA vector. The solution has the form {x1,n,M} where x1 is a particular solution of aX=b, n is the number of columns of M, and M is a matrix such that X=Mt (t is a n-vector) is the general solution of aX=0. The general solution of aX=b is x1+Mt. If the system has no integral solution, then x1 is {}. solveDiophantine[a] solves the linear diophantine system of equation ax=0 where a is a MMA matrix.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.32 smithNormalForm

smithNormalForm[mat computes the Smith Normal Form of an Alpha matrix mat and returns {u,s,v} (Alpha matrices), where u,v are unimodular and s is diagonal such that mat = u.s.v. smithNormalForm[m] computes the Smith Normal Form of a Mathematica matrix m.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.33 squareMatrixQ

squareMatrixQ[m] is True if m is a square Alpha matrix.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.34 subMatrices

subMatrices[m1,m2] subtracts two matrices m1 and m2. This function is for internal use by the Pipeline function.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.35 suppressRowNum

suppressRowNum[mat,i] suppresses row i in (MMALPHA) matrix mat.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.36 translationMatrix

translationMatrix[ind, vec] returns an Alpha translation matrix corresponding to the function: (ind → ind+vec), where ind is a list of indices and vec an integral vector.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.37 translationQ

translationQ[m] is True if the full rank affine function m is a translation, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.38 unimodularQ

unimodularQ[m] is True if an MMALPHA affine matrix m is square and unimodular, False otherwise.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.39 unimodularCompletion

unimodularCompletion[vl] completes a vector list vl into a unimodular matrix which is returned. unimodularCompletion[mat] takes the vector expressed as an Alpha matrix mat and completes it into a unimodular Alpha matrix which is returned. If the original vector is of size n, the resulting matrix is of size (n+1)*(n+1) and the first row of the resulting matrix is

the original vector with 0 appended as constant term. For example,
unimodularCompletion[{1,1,1}] returns
matrix[4, 4, {}, {{1, 1, 1, 0}, {0, 0, 0, 1}, {0, 0, 1, 0}, {0, 1, 0, 0}}]
and
unimodularCompletion[matrix[4, 4, {}, {{1, 1, 1, 0}, {0, 0, 0, 1}}]
returns
matrix[4, 4, {}, {{1, 1, 1, 0}, {0, 0, 0, 1}, {0, 0, 1, 0},
{0, 1, 0, 0}}].
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.40 unimodCompl

unimodCompl[mat] returns an unimodular completion of matrix mat, or $Failed if an error occurs. The result is an Alpha matrix. The indices of the result are those of mat, with some new arbitrary indices if necessary. Exemple : completion of (i,j→2i) returns (i,j,k→2i+k,j,i). The optional parameter indicates the dimension of the parameter space (0 if none).
unimodCompl[mat] is the mathematica version (no constant row and column). It returns a square unimodular MMALPHA matrix. First lines of this matrix are the matrix mat (modulo an extension on the right).
For exemple: {{2,0,0},{1,2,0}} will be completed as:
{{2,0,0,1,0}, {1,2,0,0,1}, {0,0,1,0,0}, {1,0,0,0,0}, {0,1,0,0,0}}.
unimodCompl[mat,nbPar] returns a MMA matrix or $Failed if an error occurs. mat is a (k*(n+p+1) MMA matrix, it represents a k-dimensional affine function for a n-dim variable, with a p-dim parameter space. Thus, the last column is the constant column. nbPar is the parameter space dimension, i.e. p. The function computes n from p and mat. The result is a MMA matrix, which has at least (n+p+1) rows and exactly (n+p+1) columns. The first k rows are exactly mat. If the result isn't square, it means that some dimension will be added by the returned change of basis.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.3.41 simplifyAffines

simplifyAffines[] simplifies all affine functions in system $result.
simplifyAffines[sys] simplifies all affine functions on system sys.
Defined in file: Alpha/Matrix.m
Package: Alpha‘Matrix‘

12.4 The Alpha‘Meta‘ package

12.4.1 directory

option of meta. If Null, the directory is Alpha. Otherwise, string which gives the directory
Defined in file: Alpha/Meta.m
Package: Alpha‘Meta‘

12.4.2 meta

Defined in file: Alpha/Meta.m
Package: Alpha‘Meta‘

12.5 The Alpha‘Options‘ package

12.5.1 debug

option (Boolean). If True, debug mode.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.2 verbose

option (Boolean). If True, print intermediate information.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.3 recurse

option (Boolean). If True, apply the function recursively to all the subsystem.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.4 library

option (List of Alpha‘system). Defines the library which must be used by the function.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.5 contextDomain

option of addLocal. If True (default), the new variable is added with the contextual domain of the expression, with the form of addLocal with a position. Otherwise, the expression domain is used.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.6 invert

option of serializeReduce. If True (default False), the nullspace vector is inverted
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.7 norm

option of removeIdEqus. If True, normalization is done
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.8 inputEquations

option of removeIdEqus. If False, an equation of the forme X = in, where in is an input variable, is not removed.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.9 allLibrary

option of removeIdEqus (default False). If set to True, all programs of library are treated.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.10 resolutionSoft

(+) resolutionSoft is an option of schedule. If resolutionSoft → pip, the schedule is computed using Pip, through a file interface (implemented in Farkas resolution only). None of the following are yet implemented: If resolutionSoft → mma, the schedule is computed using the Mathematica

function ConstrainedMin (implemented in Vertex resolution only). resolutionSoft → lpSolve the schedule is computed using the LpSolve library function ConstrainedMin (not implemented yet).
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.11 pip

possible value (default) of the options resolutionSoft of schedule
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.12 mma

possible value of the options resolutionSoft of schedule
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.13 lpSolve

possible value of the options resolutionSoft of schedule
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.14 schedMethod

Options of schedule (symbol), select the scheduling method to be used. schedMethod→farkas (default) stands for P. Feautrier's method (using PIP with file interface) implemented in the scheduleFarkas function. schedMethod→farkas2 stands for the most recent implementation of this method (ModularSchedule) .schedMethod→Vertex stands for P. Quinton's vertex method (using MMA linear resolveur). WARNING: the setting of this options greatly influence the setting of the others options of schedule because they correspond to two completely different implementations
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.15 farkas

value of options schedMethod of schedule, the schedule will be computed with P. Feautrier's method (using PIP with file interface)

Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.16 vertex

value of options schedMethod of schedule, the schedule will be computed with P. Quinton's vertex method (using MMA linear resolveur)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.17 integerSolution

(+) integerSolution is an option of schedule. If integerSolution → True (default for farkas resolution) the schedule has integral coefficients. If integerSolution → False, (default for Vertex resolution) the schedule may have rational coefficients.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.18 bigParPos

bigParPos is an option of Pip related functions, Integer type. It indicates the position of the big Paramter. If set to negative integer, there is no big parameter
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.19 addConstraints

option of schedule. The type is : List of String or List of List of String (in case of MultiDim schedule), default value : {}. Additional constraints for the LP. add constrains expressed in the strings to the current scheduling process. schedule[addConstraints→{"TA[i,j,N]=i+2j-2","TBD2==2","TBD1+2TBD3>=1"}] Impose that the schedule function of variable A is i+2j-2, Then we have constraints on the coefficients of the timing function of the B variable (the second coefficient must be 2, etc...). In case of a multi-dimensionnal scheduling, the ith list is added to the constraints of the ith dimension
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.20 durations

option of schedule (List) default {}. select a mode for counting the duration of instructions. durations → {} : each equation is 1 (whatever complex is the computation) (default). durations → ListOfInteger: The duration of each equation is given by the user the options "durationByEq" indicates whether these values stand for a duration by equation (default in Farkas resolution, not implemented in Vertex resolution) or a duration by dependence (default in Vertex resolution, not implemented in Farkas resolution). durations → ListOfListOfInteger: same as previous one but for multidimensionnal scheduling (one list of integer by dimension).
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.21 durationByEq

option of schedule (boolean) indicates whether the duration values given in the "durations" option stand for a duration by equation (default in Farkas resolution, not implemented in Vertex resolution) or a duration by dependence (default in Vertex resolution, not implemented in Farkas resolution). In the case of a duration by equation, the list must contain as many integer as there are VARIABLES (do not forget the inputs and output), the order is the order of declaration in the Alpha program. In the case of a duration by dependence, the must contain as many integer as there are dependecies the order is the order of the dependencies returned by the dep[] function
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.22 givenSchedVect

options of Schedule, List of schedule given for some variable, the syntaxe for each schedule is the one present in $schedule: {{a, {i, j, N}, sched[{0, 0, 0}, 0]}} (The list of indice may be replaced by {})
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.23 affineByVar

Value of option scheduleType of schedule, the resulting schedule will be affine by variable

Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.24 sameLinearPart

Value of option scheduleType of schedule, the resulting schedule will be affine by variable with the same linear part for all local variables
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.25 sameLinearPartExceptParam

Value of option scheduleType of schedule, the resulting schedule will be affine by variable with the same linear part for all local variables
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.26 scheduleType

scheduleType is an option of schedule (symbol) which gives the type of the schedule searched. scheduleType → affineByVar (default) : affine by variable schedule. scheduleType → sameLinearPart: affine with constant linear part. scheduleType → sameLinearPartExceptParam: affine with constant linear part except for the parameters
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.27 optimizationType

options of schedule (symbol), indicates the objective fonction used by the schedule: minimizing time, delays on dependencies of nothing. optimizationType→time tries to get aminimal time schedule (default). optimizationType→delay tries to minimize the delays on the dependencies (not implemented currently) optimizationType→Null no objective function is provided (not implemented in Vertex resolution). In that case the coefficient of the timing function are minimized lexicographically (from output to input).
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.28 time

possible value (default) of option optimizationType of schedule
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.29 delay

possible value of option optimizationType of schedule
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.30 multi

possible value of option optimizationType of schedule or of option structSchedType of buildSchedConstraints
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.31 mono

possible value of option structSchedType of buildSchedConstraint
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.32 scheduleDim

Option of buildSched Constraint (integer or list of integer), indicates the dimension to which the constraints should be set
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.33 structSchedType

Option of buildSched Constraint (integer or list of integer), indicates whether the use will be pipelined (no additionnal dimension in the schedule) or considered as an instantaneous call (additionnal dimention in the schedule)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.34 multiSchedDepth

options of schedule (integer) indicating the depth of the current schedule taking place in a multi dimensionnal scheduling process (this option should not be set by the user, it is for internal use only)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.35 onlyVar

Option of schedule which is a list of string (default value: all) indicating the only variables that we wish to schedule. Warning, if some variable are needed for the computation of the one indicated, the function will try to find their execution dates in $schedule (only implemented in the Farkas method)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.36 all

value of various options (onlyVar, onlyDep,....)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.37 onlyDep

Option of schedule which is a list of integer (default value: all) indicating the only dependences that we wish to schedule (used for multiSched[])
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.38 objFunction

objFunction is an option of schedule which gives the type of minimization done for minimizing what we try to minimize (i.e. time or delay or whatever). the technique for that is to build a function of the parameters of the system and to minimize the coefficient of this function with respect to constraints which ensure that this function is greater than what we want to minimize. The technique used by default (objFunction→lexicographic) in the Farkas resolution is to minimize lexicographically the coefficient of this function in the order of their declaration in the Alpha program. but one can chose to minimize them in another order: objFunction→lexicographic[”N”,”P”,”M”]

(not implemented anywhere) or to minimize a function of these coefficients example: objFunction→2"N"+"P" (only implemented in the vertex resolution)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.39 lexicographic

possible value of the objFunction option of the schedule function
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.40 checkSched

(+) checkSched is an option of schedule which allows a schedule to be checked for a given Alpha program. This option can be used only for affine by variable schedules. The form of this option is checkSched → list, where list has the same syntax as $schedule (see ?$schedule for more information).
WARNING: this option is not implemented
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.41 parameterConstraints

parameterConstraints is an option of dep. If set, constraints on the parameters are included in the domains of the dependencies. Default is False
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.42 subSystems

options of schedule (boolean) and dep indicating whether or not the function takes into account the calls to other systems (default, false)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.43 subSystemSchedule

option of schedule (list), indicates, if the subSystems option is set to True, the schedules of the different subsystems called in the system to schedule for performing a structured scheduling

Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.44 multiDimensional

option of Schedule (boolean), indicates if we perform a multi dimensional scheduling or mono dimensionnal scheduming (default)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.45 outputForm

options of schedule, gselect the output of the schedule which may be a schedule, a LP, a domain .... The default value is outputForm→scheduleResult i.e. the result isx a schedule (see ?$schedule). other value: outputForm→lpSolve (Linear Programming problem (LP) formated for lp_solve), outputForm→lpMMA (LP formated for MMA (to be implemented)), outputForm→domain (schedule polytope (Alpha‘domain,Warning works for small programs))
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.46 dataFlowConstantsNull

dataFlowConstantsNull is an option of scd. Its default value is True, forcing the constant part of the first level of a data-flow schedule to be 0 for all variables.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.47 dataFlowPeriod

dataFlowPeriod is an option of scd. Its default value is 1. If set to an integer, forces the period of the data-flow schedule to be this integer. If set to any non integral value, the data-flow period is choosen automatically, and assumed to be >=1. Most often used in combination with dataFlowConstantsNull set to False.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.48 dataFlowVariables

dataFlowVariables is an option of scd, which is considered only for data-flow schedules. Its default value is <<all>>, meaning that all variables are considered for data-flow scheduling. If it is set to a list of strings, it gives the list of variables which must not be inserted in the dataflow variables. These variables will be considered only for scheduling in the next dimensions of the schedule.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.49 verticesPositives

verticesPositives is an option of scd, scheduleConstraints, and timeMinSchedConstraints. Its default value is False. When set, this option forces the scheduler to find timing-functions which are positive at the vertices of the domains of the variables. Usually, only constraints on rays of the domains are taken into account. This option is not used, as it most often results in non-integral (i.e. strictly rational) schedules.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.50 sortOrder

sortOrder is an option of showSchedResult, which specifies the order in which the results are given. Default is noOrder. Other possibilities are lexicographic, or an integer which specifies the number of coefficient given backwards
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.51 noOrder

noOrder is a value for the option sortOrder of showSchedResult
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.52 eliminatesDoubles

Options of dep[], if set to False dep[] returns the usual dependence list. If set to True, the dependences returned does not contain twice the same

dependence
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.53 equalitySimpl

options of dep[], if set to True, try to simplify the dep according to the context (default True)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.54 scalarTypeCheck

option to analyze[] (Boolean). If True, perform scalar type checking. For internal use mostly.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.55 occurence

option of inlineAll[] and inlineSubsystem[] (Integer). When a given subsystem is used several times in the same caller system, the value of occurence selects the instance to be inlined.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.56 rename

option of inlineAll[] and inlineSubsystem[] (Boolean). rename → True forces the renaming of all variables, whereas if rename → False, variables are renamed only in case of conflict.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.57 renameCounter

option of inlineAll[] and inlineSubsystem[] (Integer, default 1). Sets the value to be appended to variable names in case when they are renamed to avoid name conflicts.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.58 current

option of inlineAll[] and inlineSubsystem[] (Boolean). If True, $result and $program are updated.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.59 underscore

option of inlineSubsystem and inlineAll (Boolean). When True (default), new identifier separator is _, whereas it is X otherwise
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.60 indexnorm

option of normalize[] (Boolean). indexnorm → True normalizes index expressions also. The default is False.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.61 alphaFormat

options of simplifySystem: alphaFormat→Alpha (default) simplify a system to the standard normalized Alpha form. alphaFormat→Alpha0 simplify to Alpha0 format, alphaFormat→AlpHard is not implemented
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.62 Alpha0

value of the alphaFormat options of simplifySystem
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.63 initZeroReg

options of toAlpha0v2: initZeroReg→False do not generate a control signal for the register that are initialized with a zero. These register will be initialized by the Rst signal
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.64 verifyCone

verifyCone is an option of uniformization which verifies whether the dependence cone is pointed if set to True (default value) or does not if it is False.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.65 alignInp

alignInp is an option of uniformization. If set to True the uniformize function will try to uniformize a dependence by aligning the input instead of routing the dependence. The default value is False.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.66 routeOnce

routeOnce is an option of uniformization. If set to True the uniformize function will perform routing for the first vector of the route. The default value is False.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.67 tmpFile

tmpFile is an option of uniformization. If set to True the uniformize function will output a file in /tmp directory after each iteration of uniformization. The default value is False.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.68 silent

option of show (False). If silent is True, show does not print its result, but returns a string.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.69 allVariablesAllowed

Option of changeOfBasis, default False. If True the change of basis can be applied to any variable. Not yet operational...

Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.70 showSquareDeps

option of uniformizeInfo
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.71 showNonSquareDeps

option of uniformizeInfo
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.72 showUniformDeps

option of uniformizeInfo
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.73 showNonUniformDeps

option of uniformizeInfo
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.74 showUniformizableDeps

option of uniformizeInfo
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.75 showNonUniformizableDeps

option of uniformizeInfo
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.76 minimize

option of simplex. Default is True
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.77 boundedDelay

option of timeMinSchedConstraints. Currently not used.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.78 extraEdges

Option of depGraph, allowing edges to be added
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.79 labelOffset

Option of depGraph, allowing the offset of the position of a label to be modified. Default is 0.025.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.80 labelSize

Option of depGraph, allowing the size of the labels to be changed. Default value is 0.1 .
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.81 onlyIdDep

Option of depGraph (default False), If set to True, build a graph where only identity dependences are present. (require an aligned program).
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.82 cellType

Option of a2v
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.83 tempFile

Option of a2v
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.84 vhdlLibrary

Option of a2v
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.85 compass

Option of a2v
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.86 compactCode

Option of a2v
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.87 clockEnable

Option of a2v
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.88 skipLines

Option of a2v
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.89 stdLogic

Option of a2v. If set, produces stdlogic types. Default value is True.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.90 initialize

Option of vhdlType. If set, produces initialized declarations
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.91 controler

Option value for option cellType of alphaToVHDL
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.92 cell

Option value for option cellType of alphaToVHDL
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.93 module

Option value for option cellType of alphaToVHDL
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.94 rewrite

rewite: option (boolean) for cGen. If True, output file is overwritten if it already exists.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.95 lyrtech

Option of a2v. If True, generates strange code for the Lyrtech environment. For example, rst is not a std_logic but a vector of std_logic of size 1.

Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.96 noPrint

boolean option of cGen. If True, the generated C code does not provide prompts before the inputs, nor before the outputs, which makes it easier to run a test from an input file
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.97 alreadySchedule

option of cGen (boolean, default false), if set to True, cGen does not perform schedule neither change of Basis: WARNING not Implemented yet
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.98 stimuli

option of cGen (boolean, default False), if set to True, cGen generates one stimuli file by input and output variable
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.99 interactive

interactive is a (boolean) option of cGen. If True, cGen also generates a function main which (i) reads system inputs on stdin (ii) evaluates the system, (iii) prints the outputs on stdout
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.100 matlab

matlab is a boolean option of cGen. If True, cGen also generates a function mexFunction which is used for interface with matlab code
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.101 bitTrue

bitTrue is a boolean option of cGen. If True, cGen includes the bitOperators.h file in the C code and produces the compile script to compile the code
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.102 debugC

debugC: option (boolean) for VirtexGen. If True, Virtexgen generates a function main which (i) reads system inputs on stdin (ii) evaluate the system, (iii) prints the outputs on stdout (the output are not significant in that case)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.103 onlyLocalVars

option used when a function shouyld be applied sometimes nly on local variables
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.104 projVector

options of appSched to indicate what is the projection vector (appSched will try to find a unimodulary completion that is a projection along this vector) should be a Mathematica Vector (integer list) for instance: {1,0,0}(dimension is the number of indices plus parameter). Warning, this function does not try very hard to find a projection, the first one found is choosen. If it failed, please use the projMatrix options
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.105 projMatrix

options of appSched to indicate what is the projection Matrix used for completing the schedule. should be a Mathematica Matrix (list of list of integer) for instance: {{1,0,0},{0,1,0}}. This matrix is a linear Matrix, i.e. it does not contains the affine part information (dimension is the number of indices plus parameter)
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.106 checkCase

checkCase is an option of getContextDomain. If set, the function checks on the fly that the branches of the case expressions do not overlap. False by default.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.107 mergeDomains

mergeDomains is an option of simplifySpaceDom. If set (True), the domains are merged before the simplification is applied. Default value is False
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.108 structured

structured is an option of alpha0ToAlphard which specifies that a program may contain subsystems
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.109 onlyModules

onlyModules is an option of alpha2ToAlphard. If set (True), the program does not accept a system which is not a module. Default is False.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.110 startTime

startTime is an options of systemC generator for controleurs to indicate the starting time of an Alphard program, i.e. the smallest value taken by the t index upon all the domains of the library, default Infinity
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.111 systemCFiles

systemCFile is an options of systemC generator that indicates whether the codeis generated in one .h file (option value 1, default value) or in two files (.h and .cpp file, option value 2)

Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.112 busWidth

options of socLibGen (default 32) set the bus Width used during the generation of the systemC interface to Alphard programs
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.113 inputOrOutput

options of socLibGen for internal use
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.114 exceptions

option of several functions. Gives a list of variables or subsystems that has not to be considered for space time decomposition
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.115 remIdDeps

option of simplifySystem, false by default. If set, all identity dependences are removed. If it appears that this simplification is safe, It’ll become a default option. PQ. July 2004.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.5.116 remIdEqus

option of simplifySystem, false by default. If set, all identity equations are removed, unless the format is alpha0. If it appears that this simplification is safe, It’ll become a default option. PQ. July 2004.
Defined in file: Alpha/Options.m
Package: Alpha‘Options‘

12.6 The Alpha‘Properties‘ package

Documentation checked on August 8, 2004

12.6.1 Properties

The Alpha‘Properties‘ package contains the definition of various property-checking functions. Should be used as the preferred place for adding new property-checking predicates that might be of large interest.
Defined in file: Alpha/Properties.m
Package: Alpha‘Properties‘

This package should probably merged with other packages such as Tables, Substitution, etc.

12.6.2 allDomEqualQ

allDomEqualQ[domList] is True is all the domains in the list domList are equal.
Defined in file: Alpha/Properties.m
Package: Alpha‘Properties‘

12.6.3 allDomDisjointQ

allDomDisjointQ[domList] is True if all the domains in a list domList are two-by-two disjoint.
Defined in file: Alpha/Properties.m
Package: Alpha‘Properties‘

12.6.4 allDomUnion

allDomUnion[domList] returns the domain which is the union of the domains in domList.
Defined in file: Alpha/Properties.m
Package: Alpha‘Properties‘

12.6.5 uniformQ

uniformQ[sys] is True if the Alpha program sys is uniform (all dependences between local variables are translations), False otherwise. The default value

of sys is $result. uniformQ can also be applied to an Alpha matrix or to an affine AST.
Defined in file: Alpha/Properties.m
Package: Alpha‘Properties‘

12.7 The Alpha‘Reduction‘ package

Documentation revised on August 8, 2004

12.7.1 Reduction

The Alpha‘Reduction‘ package contains transformations to deal with reduction operators.
Defined in file: Alpha/Reduction.m
Package: Alpha‘Reduction‘

This package is not completed. In particular, it contains two functions splitReduction and splitReduce which do no work.

12.7.2 serializeReduce

serializeRecuce[pos] serializes the reduction at position pos in $result. serializeReduce[pos, spec] serializes the reduction at position pos in $result using serialization specifier spec (spec is a string). Parameter pos may be either a position in the AST, or the name (string) of a variable whose definition has the form "pos = reduction". serializeReduce[sys, pos, spec] does the same to program sys. Parameter spec is a string containing the new variable name composed with the new serialized dependence (parameters need not be given explicitly). For example: "Z.(i,k→i,k-1)" means that the serialized variable will be "Z" and the serialize direction will be the vector (0,-1). If no spec is given, the function guesses the direction of serialization, and the option (invert → False|True) allows this direction to be changed. The guess is based on the null space of the reduction function, and it works only if this null space has exactly one vector.
Defined in file: Alpha/Reduction.m
Package: Alpha‘Reduction‘

12.7.3 isolateReductions

isolateReductions[] locates all the reductions in a system and isolates these reductions in new equations
Defined in file: Alpha/Reduction.m
Package: Alpha‘Reduction‘

12.7.4 isolateOneReduction

isolates one reduction
Defined in file: Alpha/Reduction.m
Package: Alpha‘Reduction‘

12.7.5 splitReduction

splitReduction[y] tries to rewrite the reduction at rhs of symbol y as a sequence of reduction. This function does not work currently.
Defined in file: Alpha/Reduction.m
Package: Alpha‘Reduction‘

12.8 The Alpha‘Schematics‘ package

12.8.1 Schematics

This package contains functions for drawing a raw schematics of an Alpha programs
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.2 partShown

partShown is an option of schematics. It defines the region of the plot that will be shown. Default is Automatic and corresponds to a region {{0,1},{0,1}}, which displays the full schematics. partShown →{{0.2,0.4},{0.3,1.2}} would display a region whose coordinates, relative to the schematics, are changed.
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.3 fSize

fSize is an option of schematics, which changes the size of the font used to show symbols. Default value is 20 .
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.4 sPositions

sPositions is an option of schematics. Default value is square. sPositions → Null stacks the operators. sPositions → {columns → nb} gives the number of columns of the diagram. sPositions may be a list of coordinates for the operators. In this case, make sure that the number of positions is the same as the number of operators.
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.5 square

square is an option value for option sPosition in the schematics function. sPosition → square asks for a square shaped schematics layout.
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.6 columns

columns is an option value for option sPosition in the schematics function. sPositions→{columns→nb} asks for a nb column schematics layout.
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.7 yFactor

yFactor is the form factor of boxes used by the schematics function. Default value is 0.2 .
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.8 offsetX

offsetX is the X offset value for schematics. Default value is 0.5 .
Defined in file: Alpha/Schematics.m

Package: Alpha‘Schematics‘

12.8.9 offsetY

offsetY is the Y offset value for schematics. Default value is 0.1 .
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.10 newName

newName[sys, v] returns a new unique name built from variable v. newName[v] does the same in $result. This function cannot be used in any context. Use getNewName instead.
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.11 skeleton

skeleton[] gives the skeleton of $result
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.12 schematics

schematics[] draws a schematic diagram of the program $result. See Options[schematics] to get the options of schematics.
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.13 flattenEquation

Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.14 flattenSkeleton

ddddd
Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.8.15 findAliases

Defined in file: Alpha/Schematics.m
Package: Alpha‘Schematics‘

12.9 The Alpha‘Semantics‘ package

Documentation revised on August 10, 2004

12.9.1 Semantics

Alpha‘Semantics‘ is the package which contains the functions for computing the type of Alpha expressions. These functions are changeType, expDimension, expDomain, expType, getContextDomain, matchTypes, and replaceByEquivExpr.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

12.9.2 changeType

changeType[sys,var,newType] changes the type of the variable var to newType in sys (default $result). No typing compatibility check is done: this is not a semantic preserving transformation. Mainly used for changing integers to fixed bit width integers.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

12.9.3 expDimension

expDimension[sys,exp] computes the dimension of the expression exp the in system contained in the system sys (default $result). The expression may be given as a string, a position vector (following the convention of the Mathematica function Position) in the system, or as an AST. expDimension outputs error messages if the dimensions are not compatible, and returns dimension -1 in this case. Remark: parameters are counted as additional dimensions.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

12.9.4 expDomain

expDomain[sys,exp] computes and returns the domain of exp in program sys (default $result) or $Failed if the expression is badly formed. expDomain generates error and warning messages accordingly. The expression may be given as a string, a position vector (following the convention of the Mathematica function Position) in the system, or as an AST.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

12.9.5 expType

expType[sys,exp] computes the type (integer, boolean, real, bottom) of an expression and checks its correctness using type–matching rules. Default value of sys is $result. The expression may be given as a string, a position vector (following the convention of the Mathematica function Position) in the system, or as an AST. expType returns bottom if the expression is not correctly typed. In such a case, the offending tree and the colliding types are printed on the screen.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

12.9.6 getContextDomain

getContextDomain[sys,pos] computes the context domain in which an expression at position pos in sys is used. Default value of sys is $result. The context is the domain inherited by an expression from the constructs which surrounds it. The position is counted from the root of the system following the Mathematica notion of position.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

12.9.7 matchTypes

matchTypes[type1,type2] computes the type corresponding to the union of those of the parameters (integer, boolean, real) and returns the resulting type or bottom if the types are incompatible.
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

12.9.8 replaceByEquivExpr

replaceByEquivExpr[sys,pos,expr] replace in sys the expression present at position pos by expr. Default value of sys is $result. replaceByEquivExpr[sys,expr1,expr2] replace in sys all occurences of expr1 by expr2. expr1 and expr2 can be given in AST form of Alpha form. Warning: the equivalence of the resulting program is NOT guaranteed. A test is made to check if both expressions are equivalent, but if this test fails, the replacement is done anyway. This function should be used for replacing an expression by an equivalent one when this cannot be done automatically (for instance, in presence of degenerated domains).
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

12.9.9 setBitWidth

setBitWidth[var:_String,value:_Integer] change the scalar type of variable var1 in system sys if this variable has a specified bitwidth (e.g. integer[S,5]) and set it to the bitwidt value, if the variable does not have a specified bitwidth, it does nothing and issue a warning
Defined in file: Alpha/Semantics.m
Package: Alpha‘Semantics‘

12.10 The Alpha‘Static‘ package

Documentation revised on August 10, 2004

12.10.1 Static

The Alpha‘Static‘ package contains functions for the static analysis of Alpha programs. These functions are: analyze, dep, and checkUseful.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

12.10.2 analyze

analyze[sys] performs a static analysis of system sys (default $result) and returns True if the analysis is successful (even with warnings), False otherwise.

The analyze function has options verbose, recurse, library and scalarTypeCheck (see Options[analyze]). Option verbose (Boolean; default True) reports analysis progress. Option recurse (Boolean; defaut False) recursively looks up for all the subsystems involved and analyzes them, too. Option library (List; default $library) gives the library to search for subsystems. Example of option usage: analyze[recurse→True, verbose→False]
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

12.10.3 dep

dep[sys] generates a dependency table for sys (default $result). The form of the table is:
dtable[{
depend[Domain, LHS_var_name, RHS_var_name, Matrix, RHS_var_domain],
depend[Domain, LHS_var_name, RHS_var_name, Matrix, RHS_var_domain],
. . .
].
The table may be pretty–printed using show[dep[]].
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

12.10.4 dep1

dep1[sys, {occur:{_Integer..}, occurs___}] generates dependency table entries for all occurrences in the list. Used by dep[] to generate the dependency table.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

12.10.5 checkUseful

checkUseful[sys, var] checks that the variable var is used in the system sys for all the points of its domain. Default value of sys is $result. If var is omitted, the function is applied to all variables of sys.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

12.10.6 checkDeclarations

This function is a private function of the package.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

12.10.7 buildLHSIdList

This function is a private function of the package.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

12.10.8 buildRHSIdList

This function is a private function of the package.
Defined in file: Alpha/Static.m
Package: Alpha‘Static‘

12.11 The Alpha‘SubSystems‘ package

Documentation revised on August 10, 2004

Bugs: test[SubSystems] fails. inlineAll should test that $result contains a system.

12.11.1 SubSystems

The Alpha‘SubSystems‘ package contains functions to operate on Alpha subsystems. These functions are assignParameterValue, assignParameterValueLib, fixParameterValue, inlineAll, inlineSubSystem, spread, removeIdEqus, simplifyUseInputs, substDom, subSystemUsedBy, and topoSort.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.2 addParameterId

function externalized for debugging purposes.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.3 affExtHom

function externalized for debugging purposes.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.4 assignParameterValue

assignParameterValue[param,v,sys] gives the value v to the parameter param in the Alpha system sys and returns the modified system. param is a string, and v an integer. Default value of sys is $result.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.5 assignParameterValueLib

see fixParameter.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.6 fixParameter

fixParameter[param,v,systName] gives the value v to a parameter param in an Alpha system of name ”systName” and in the call to ”systName” in $library. It returns the new library (list of Alpha systems) and modifies $library (except if $library is explicitely specified as the 4th parameter). If the system name is ommited, the function sets the parameter in all the programs of $library and in $result. WARNING: currently, this function supposes that the parameter ”param” has the same value everywhere, you also have to ensure that $result is the top calling system of the library (i.e. the module in an AlpHard program).
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.7 inlineAll

inlineAll[options] inlines all the subsystems of system $result. Options are rename, renameCounter, verbose, caller, library, and current. (see Options[inLineAll] for default values).
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.8 inliningRenameCounter

Variable. Holds the counter value used to avoid name conflicts while renaming the variables.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.9 inlineSubsystem

inlineSubsystem[name,options] searches the current system (see option : caller) for a use statement of the subsystem name, and replaces this use statement with its definition extracted from $library, with proper variable renaming and parameter instanciation. Returns the modified system if no error, the caller otherwise. Side effect: if the "current" option is set, it sets $program to the previous Alpha‘$result and sets $result to the returned value. Options are occurence, rename, renameCounter, verbose, caller, library, undescore, and current (default options in Options[inlineSubsystem]).
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.10 library

library is an option of inlineAll and inlineSubsystem. library → list of systems (default $library) specifies the list of systems to search for a subsystem. When more than one declaration of the same system appear in the library, the first one is inlined.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.11 occurence

Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.12 underscore

option of inlineSubsystem and inlineAll (Boolean). When True (default), new identifier separator is _, whereas it is X otherwise.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.13 spread

spread[var, index] replaces var in system $result by a set of variables varI, where I is in the range of index of var. This function is not completed and should not be used.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.14 removeIdEqus

removeIdEqus[sys] removes in sys the equations of the form A=B as introduced for example by the inlineSubsystem and inlineAll transformations. Warning, this function does not normalize the resulting system, unless option norm → True is used. removeIdEqus has options norm, allLibrary and inputEquations.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.15 simplifyUseInputs

simplifyUseInputs[sys] adds local buffer variables for each input which is not already a simple variable (input to a use may be any expression). The new variable is defined on the 'real domain' of the expression: its context domain intersected with its use domain.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.16 substDom

substDom[dom, extDom, paramAssign, nbparams] computes the transformation of a domain dom in a subsystem inlining. For internal use only.
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.17 subSystemUsedBy

subSystemUsedBy[sys] returns the list of names of subsystems used by system sys (default $result).
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.11.18 topoSort

topoSort[lib] returns the list of system names of library lib, sorted in the reverse hierachical order, i.e. if a system A uses a system B and a system C, topoSort returns {B, C, A} or {C, B, A}
Defined in file: Alpha/SubSystems.m
Package: Alpha‘SubSystems‘

12.12 The Alpha‘Tables‘ package

Documentation revised on March 11, 2008

12.12.1 Tables

The Alpha‘Tables‘ package contains variable and function definitions relative to the retrieval of contextual information on nodes and programs. Functions are: addAllParameterDomain, addParameterDomain, getDeclaration, getDeclarationDomain, getDefinition, getDimension, getEquation, getIndexNames, getInputVars, getLocalVars, getOutputVars, getVariables, getSystemName, getSystemParameters, getSystemParameterDomain, symToString, lookUpFor, lookUpForPos, accessByPath, and undoModif.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.2 addAllParameterDomain

addAllParameterDomain[sys] adds the constraints present in the parameter domain of system sys to each domain of the system. addAllParameterDomain[] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.3 addParameterDomain

addParameterDomain[dom, paramdom] adds to the constraints of dom the constraints on the parameters defined in paramdom. For internal use mostly.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.4 getDeclaration

getDeclaration[var] returns the complete declaration of variable var in system $result or an empty list if the declaration does not exist. getDeclaration[sys, var] returns the declaration of var in system sys.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.5 getDeclarationDomain

getDeclarationDomain[v] returns the declaration domain of variable v in system $result. getDeclarationDomain[sys,v] returns the declaration domain of variable v in system sys
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.6 getDefinition

getDefinition[var] returns the RHS of the equation defining variable var or an empty list if the definition does not exist. If the variable is defined as output of a subsystem, returns an empty list and prints an error message.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.7 getDimension

getDimension[dom] returns the dimension of the Alpha domain dom.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.8 getEquation

getEquation[var] returns the equation defining a variable var or an empty list if the definition does not exist. If the variable is the output of a subsystem, getEquation issues a warning and returns the use statement of this subsystem.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.9 getIndexNames

getIndexNames[sys, var] returns the list of index names from a variable's declaration. getIndexNames[var] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.10 getInputVars

getInputVars[sys] returns the list of input variables used in system sys. getInputVars[] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.11 getLocalVars

getLocalVars[sys] returns the list of local variables used in system sys. getLocalVars[] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.12 getOutputVars

getOutputVars[sys] returns the list of Output variables used in system sys. getOutputVars[] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.13 getVariables

getVariables[sys] returns the list of variables used in system sys. getVariables[] does the same to $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.14 getSystemName

getSystemName[sys] return the name of system sys. getSystemName[] gives the name of $result.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.15 getSystemParameters

getSystemParameters[sys] return the parameters of the system sys (default $result)
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.16 getSystemParameterDomain

getSystemParameterDomain[sys] return the parameters domain of the system sys (default $result)
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.17 symToString

symToString[symb_] Converts a symbol to string. Leaves any other object simply evaluated. Returns the string containing its name according in current output format.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.18 lookUpFor

lookUpFor[sys, position:{_Integer...}, operator_Symbol] returns the smallest tree whose root is the specified by operator and which contains the specified position, or an empty list if operator not found.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.19 lookUpForPos

lookUpForPos[sys, position:{_Integer...}, operator_Symbol] returns the position of the nearest occurrence of a specific operator surrounding a specified position, or an empty list if operator not found.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

12.12.20 accessByPath

accessByPath[path:{_Integer..}] extracts the part of $result specified by subtree position path. accessByPath[tree_, path:{_Integer..}] extracts the part

of a tree specified by its subtree position specifier path
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

Only one position can be supplied at a time, so a single tree is returned if the position is valid (no check is done.)

12.12.21 undoModif

undoModif[] called after a program transformation undoes its effects. Returns the program as before the last transformation.
Defined in file: Alpha/Tables.m
Package: Alpha‘Tables‘

’undoModif’ copies the contents of ’Alpha‘$program’ into ’Alpha‘$result’ so the result of the last transformation (see description of ’Alpha‘$result’ and ’Alpha‘$program’) is reset to the result of the previous one.

12.13 The Alpha‘Visual‘ package

Documentation revised on August 10, 2004

12.13.1 Visual

The Alpha‘Visual‘ packages contains a few functions to visualize 1D or 2D domains. The main function is showDomain. Function getBoundingBox is also used in some other packages, as well as function bbDomain, which is poorly written.
Defined in file: Alpha/Visual.m
Package: Alpha‘Visual‘

12.13.2 bbDomain

bbDomain[d] returns a list representing the bounding box of domain d. This has the form {{xmin,xmax},{ymin,ymax},{xneg,xpos}, {yneg,ypos}} where {xmin,xmax},{ymin,ymax} is the bounding box of the vertices, {xneg,xpos}, {yneg,ypos} indicate that the domain is infinite in one of these directions. If xneg and xpos are 0, no rays. If xneg is 1, negative ray, if yneg is 1, positive ray. Same for yneg and ypos

Defined in file: Alpha/Visual.m
Package: Alpha‘Visual‘

12.13.3 getBoundingBox

getBoundingBox[dom] gives the bounds of the smallest rectangle containing dom. This function is temporary, in particular it does not handle domains that are union of polyhedra (in fact it work only on convex polyhedra. It should be reimplemented using DomProject).
Defined in file: Alpha/Visual.m
Package: Alpha‘Visual‘

12.13.4 showDomain

showDomain[d] displays a plot of any 1D or 2D domain or any list of 1D or 2D domains. showDomain[d, name] displays d with the title name specified.
Defined in file: Alpha/Visual.m
Package: Alpha‘Visual‘

12.14 The Alpha‘Visual3D‘ package

Documentation revised on August 10, 2004

12.14.1 Visual3D

The Alpha‘Visual3D‘ package contains a few functions to visualize and animate 3D domains. The main function is vshow, other functions are facets, and listPlanes. Some functions are exported only for debugging purposes.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.2 facets

facets[dom] computes the list of polygons corresponding to the domain dom.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.3 listPlanes

listPlanes[l,p] returns the pair {l,lp} where lp is the lists of planes in p which contain point l.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.4 maxR

maxR is an option of vshow. It allows the viewpoint of the final picture to be set. See stepR for more details.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.5 minR

minR is an option of vshow. It allows the r parameter of the viewpoint to be changed. By default, minR is 1. See also stepR and maxR, and ViewPoint of Mathematica.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.6 orderPolygon

only here for debugging purposes.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.7 stepR

stepR is an option of vshow. It allows the step of the r parameter of the viewpoint to be changed, when one wants the domains to be animated. By default, stepR is 1. Combined with minR and maxR, it allows a sequence of pictures with viewpoint between minR and maxR, by steps stepR to be drawn.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.8 threeDDomainQ

only here for debugging purposes.
Defined in file: Alpha/Visual3D.m

Package: Alpha‘Visual3D‘

12.14.9 twoDDomainQ

only here for debugging purposes.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.10 vp1

vp1 is an option of vshow, allowing the first parameter of ViewPoint to be changed. Default value is 3.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.11 vp3

vp3 is an option of vshow, allowing the third parameter of ViewPoint to be changed. Default value is 1.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.12 vshow

vshow[d] shows a 2D or 3D graphic picture of the 3D domain d. vshow[var] shows the domain of variable var in $result (see Options[vshow] for more details).
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.14.13 units

only here for debugging purposes.
Defined in file: Alpha/Visual3D.m
Package: Alpha‘Visual3D‘

12.15 The Alpha‘Zpol‘ package

Documentation revised on August 10, 2004

12.15.1 Zpol

The Alpha‘Zpol‘ package contains additional functions for computing with Z-polyhedra. These functions are expressLatticeWithMat, expressZpolWithMat, getMatOfZpol, and getPolOfZpol.
Defined in file: Alpha/Zpol.m
Package: Alpha‘Zpol‘

The internal representation of a Zpol is: domain[dim_Integer, {___String}, {___zpol[m_matrix,{___pol}]}] (which is m(p)).

12.15.2 expressLatticeWithMat

expressLatticeWithMat[sys,m] returns a system where all Z-polyhedra using lattices corresponding to the matrix m are rewritten so as to appear with matrix m as lattice. Default value of sys is $result. expressLatticeWithMat[sys,m,pos] modifies only the Z-domain at position pos of the Alpha abstract syntax tree.
Defined in file: Alpha/Zpol.m
Package: Alpha‘Zpol‘

12.15.3 expressZpolWithMat

expressZpolWithMat[z,M] attempts to change the representation of the Z-polyhedron z. Assuming z = N(P), we expect a result of the form w = M(P') where M is the matrix given as second argument, hence it returns M(P') where P'=M$\hat{}${-1} N(P). If M and N do not represent the same lattice, it gives a warning and return z unchanged. If z is a union of Z-polyhedra, it fails.
Defined in file: Alpha/Zpol.m
Package: Alpha‘Zpol‘

12.15.4 getMatOfZpol

getMatOfZpol[z] returns the matrix used for identifying the lattice of a Z-polyhedron z. If z=M(P), it returns M. If Z is already a polyhedron, it returns the identity matrix. If z is a union of Z-polyhedra, it fails.
Defined in file: Alpha/Zpol.m
Package: Alpha‘Zpol‘

12.15.5 getPolOfZpol

getPolOfZpol[z] returns the polyhedron used as source for generating a Z-polyhedron z. If z=M(P), it returns P. If z is already a polyhedron, it returns z. If z is a union of Z-polyhedra, it fails.
Defined in file: Alpha/Zpol.m
Package: Alpha‘Zpol‘

Index