COCONUT Deliverable D14

Set of combination algorithms for state of the art modules

EPFL – *Swiss Federal Institue of Technology – Lausanne*
IRIN – *University of Nantes*
University of Vienna

July 30, 2003

Chapter 1

Management Modules

Introduction

A management module implements a mean to access and modify (in a secured and allowed way) the *search graph*. The *search graph* actually represents the search space and provides a means of navigation in this search space.

The search space is a collection of *search nodes* that contains the information produced by the inference engines. There are two types of search nodes :

full nodes that provide a complete specification of some subproblem in the problem under consideration

delta nodes that store the difference between some two subproblems

A full node consists of a *model* that explicitly specifies some subproblem and an *annotation* that stores some information about this subproblem.

A delta node is a list of so called *deltas* that specify the changes (to be done) to some full node. There are pre-defined types of deltas for specifying symbolic transformations (changes to the subproblem part of full nodes) and modifications of the annotation part of full nodes.

Hereafter are detailed different management modules that have been implemented until now.

1.1 Box chooser

1.1.1 Copyright

This information is copyright by Dr. Hermann Schichl from the Mathematics Department of the University of Vienna.

1.1.2 Module name

Module name is `Box Chooser`

1.1.3 Type of module

`Box Chooser` is a management module.

1.1.4 Description of the module

`Box Chooser` is a module which chooses a new work node from the search graph.

1.1.5 Limits

There are no hardcoded limits.

1.1.6 Service *choose*

`Box Chooser` chooses one of the leafs of the search graph as new work node, extracts the work node and updates the search focus accordingly.

Kind of input

Search graph, search database.

Output provided

The code returns 0 if a box was chosen and 1 if no box is left in the search graph.

Control parameters

`method`: This parameter is of type `int` specifies the method used to determine the next box.

0. Choose the box randomly.
1. Choose the box with biggest `log_vol`.

1.2 Delta management

1.2.1 Copyright

This information is copyright by Dr. Hermann Schichl from the Mathematics Department of the University of Vienna.

1.2.2 Module name

Module name is `Delta Management`

1.2.3 Type of module

`Delta Management` is a management module.

1.2.4 Description of the module

`Delta Management` is a module which performs various management tasks for the deltas in a work node

1.2.5 Limits

There are no hardcoded limits.

undo, undo last

1.2.6 Service *compress*

`Delta Management` collects all deltas of a work node of a specific type and compresses them into one delta performing the same change. Until now this is only implemented for the `bound_delta` type.

Kind of input

Any work node, the search database

Output provided

The return value is always 0.

Control parameters

`type`: This parameter is of type `string` and specifies the class of deltas on which the change is performed.

`max gainfactor`: With this parameter of type `double` for compression of `bound_delta`s the maximum allowed gainfactor for changes in a single variable are specified. I.e., setting `max gainfactor` to 0.5 would ignore all changes in single components which do not at least halven the width of the node range. The default is `1.`.

1.2.7 Service *undo*

`Delta Management` unapplies all deltas of a work node of a specific type.

Kind of input

Any work node, the search database

Output provided

The return value is always 0.

Control parameters

`type`: This parameter is of type `string` and specifies the class of deltas on which the change is performed.

1.2.8 Service *undo last*

`Delta Management` unapplies the last n deltas of a work node of a specific type.

Kind of input

Any work node, the search database

Output provided

The return value is always 0.

Control parameters

`type`: This parameter is of type `string` and specifies the class of deltas on which the change is performed.

`n`: With this parameter of type `unsigned int` the number of deltas to be unapplied is specified. The default is `1`..

1.3 Focus management

1.3.1 Copyright

This information is copyright by Dr. Hermann Schichl from the Mathematics Department of the University of Vienna.

1.3.2 Module name

Module name is `Focus Management`

1.3.3 Type of module

`Focus Management` is a management module.

1.3.4 Description of the module

`Focus Management` is a module which manipulates the focus of the search graph.

1.3.5 Limits

There are no hardcoded limits.

1.3.6 Service *focus*

`Focus Management` initializes a new focus.

Kind of input

Any work node, the search focus, the search graph, the search database

Output provided

The return value is 0 if there is a node to which the focus can be initialized and 1 otherwise.

Control parameters

None.

1.3.7 Service *kill focus*

`Focus Management` removes the node the focus is pointing to from the search graph and destroys the focus.

Kind of input

Any work node, the search focus, the search graph, the search database

Output provided

The return value is 0.

Control parameters

None.

1.3.8 Service *promote focus*

`Focus Management` promotes the node the focus is pointing to to its parent node and moves the focus accordingly.

Kind of input

Any work node, the search focus, the search graph, the search database

Output provided

The return value is 0 if promoting was successful and 1 otherwise (i.e., if the parent has more than one child).

Control parameters

None.

1.4 Full model management

1.4.1 Copyright

This information is copyright by Dr. Hermann Schichl from the Mathematics Department of the University of Vienna.

1.4.2 Module name

Module name is `Full Model Management`

1.4.3 Type of module

`Full Model Management` is a management module.

1.4.4 Description of the module

`Full Model Management` is a module which updates the model in the work node absolutely. This means that all changes and deltas performed in the work node are written into the model specification, hereby generating a work node which can be stored as `full_node` into the search graph.

1.4.5 Limits

There are no hardcoded limits.

1.4.6 Service *model update*

`Full Model Management` writes all DAG updates, semantics changes, and the bound changes into the model DAG and stores all annotation deltas in the search database.

Kind of input

Any work node, the search database.

Output provided

The return value is always 0.

Control parameters

None.

1.5 Solution management

1.5.1 Copyright

This information is copyright by Dr. Hermann Schichl from the Mathematics Department of the University of Vienna.

1.5.2 Module name

Module name is `Solution Management`

1.5.3 Type of module

`Solution Management` is a management module.

1.5.4 Description of the module

`Solution Management` is a module which manages the `solution` table in the search database.

1.5.5 Limits

There are no hardcoded limits.

1.5.6 Service *add*

`Solution Management` stores the box represented by the work node in the `solution` table of the search database.

Kind of input

Any work node, the search database, possibly the search graph.

Output provided

The return value is 0.

Control parameters

None.

1.5.7 Service *add/remove*

`Solution Management` stores the box represented by the work node in the `solution` table of the search database and removes the node from the search graph.

Kind of input

Any work node, the search database, possibly the search graph.

Output provided

The return value is 0 if the node was removed and if it was not the last child of its parent. The return value is 1 if the removed node was the last child of its parent.

Control parameters

None.

1.5.8 Service *purge*

`Solution Management` removes all boxes from the `solution` table, whose lower bound of the objective function is bigger than a given threshold.

Kind of input

The search database.

Output provided

The return value is 0.

Control parameters

`threshold`: This parameter is of type `double` specifies the threshold.

1.6 Split management

1.6.1 Copyright

This information is copyright by Dr. Hermann Schichl from the Mathematics Department of the University of Vienna.

1.6.2 Module name

Module name is `Split Management`

1.6.3 Type of module

`Split Management` is a management module.

1.6.4 Description of the module

`Split Management` is a module which performs splits on the work node and stores the generated nodes in the search graph. It is possible to update the work node and the focus in such a way that one of the splits is chosen as new work node.

1.6.5 Limits

There are no hardcoded limits.

1.6.6 Service *split*

`Split Management` performs one of the proposed splits to the work node and stores all generated nodes in the search graph.

One of the splits can be kept to update the work node and reset the focus to the updated work node.

Kind of input

Any work node, search focus, search graph.

Output provided

The function returns 0 if a split was performed and -1 if no suitable split could be found.

Control parameters

`transaction`: This parameter is of type `unsigned int` and specifies the entry number in the proposed splits map of the work node for the split to be chosen.

`max splits`: With this parameter of type `int` the maximum number of allowed splits is specified. The default is `INT_MAX`.

`min splits`: With this parameter of type `int` the minimum number of allowed splits is specified. The default is `1`.

`min gain`: This parameter of type `double` specifies the minimum gain in box size in every produced box. The default is `1`.

`keep`: Using this `bool` the user can specify whether one of the splits ought to be kept as new work node. Default: `false`.

`keep type`: This `int` codes which algorithm is used to determine which node to keep if `keep` is `true`:

`SPLIT_MGMT_KEEP_FIRST` (0): The first node is kept.

`SPLIT_MGMT_KEEP_BIGGEST` (1): The node with biggest `log_vol` is kept.

- **`SPLIT_MGMT_KEEP_SPEC` (2):** The n–th generated node is kept, where n is determined by the parameter `keep idx`.

`keep idx`: This `usigned int` specifies which node should be kept if `keep type` is `SPLIT_MGMT_KEEP_SPEC`.

1.6.7 Service *clear splits*

`Split Management` clears the list of proposed splits in the work node.

Kind of input

Any work node, search focus, search graph.

Output provided

The function returns 0.

Control parameters

None.