CHAPTER

ONE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

1.1 The Compiler Module:

class Compiler.**Compiler**

The **Compiler** will be an array of instructions that will later be use to generate codes in the different languages supported

createPotentialClique(*cliq*, *varPot*)

The CPO instruction create the variable used for every potential mentioned in the code

addVariablePotential(*var*, *cliq*)

Instructions for the language to add tha *var* in the *cliq* potential. Note that all these arguments are given as strings in order to generate a code later

addSoftEvidencePotential(*evid*, *cliq*, *index*, *value*)

Support for hard evidences will soon be implemented. .. note:: Evidences should be given as python dictionaries : {'a likelihood' : [its values], etc}

fillPotential(*cliq*, *value*)

Used to get all the values of a potential to 1 or 0, useful for the proper initializations in the different languages

multiplicationCPT(*cliq*, *cpt*, *varPot*)

MUC directive asks to compute a clique potential with all the probabilities of its variables

multiplicationPotentials(*cliq1*, *parcliq2*, *varPot1*, *varPot2*)

Instruction to compute two potentials : the variables of the potentials are passed as arguments for use in the Generators

marginalization(*bn*, *cliq1*, *selloncliq2*, *varPot1*, *varPot2*)

Marginalization of a clique, given another. Variables of used cliques are passed as parameters

normalisation(*cliq*, *targ*)

Instruction to normalize a clique, the last stage before the output

Compiler.**labelPotential**(*jt*, *c*)

Get the name of the potential for a clique c

Compiler.**creationPotentialsAbsorp**(*bn*, *jt*)

Fill the compiler array of instructions in order to create the potentials and add the corresponding variables

Compiler.**initPotentialsAbsorp**(*bn*, *jt*)
Instructions for the initialization of the potentials

Compiler.**creaIniOnePotDif**(*bn*, *jt*, *ca*, *cb*)
Create a potential (for ca) for the diffusion

Compiler.**creaIniOnePotTar**(*bn*, *jt*, *ca*)
Create a potential which contains a target for the diffusion

Compiler.**creaIniPotentialsDiffu**(*bn*, *jt*, *diffu*, *targets*)
Create all potentials for the diffusion, we must call this function before the absorption, and after the initialization of absorption's potentials

Compiler.**labelPotentialEvs**(*bn*, *evs*)
Returns a list of every nodes of the BN who contains an evidence

Compiler.**evsPotentials**(*bn*, *jt*, *evs*, *diffu*)
Instructions to create, fill and initialize the potentials of soft evidences. Hard evidences support will soon be implemented

Compiler.**neighbors**(*jt*, *c*)
List of all the direct neighbors of a clique c in a junction tree jt

Compiler.**isTarget**(*bn*, *jt*, *target*, *n*)
Verifies if a clique n contains a target

Compiler.**mainClique**(*bn*, *jt*, *target*)
Gives the main clique of a junction where the information will be focused

Compiler.**parcours**(*bn*, *jt*, *targetmp*, *n*, *r*, *absorp*, *diffu*)
Returns two lists for the absorption and the diffusion of the information in the junction tree

Compiler.**labelSeparator**(*jt*, *ca*, *cb*)
Returns the separator's label between ca and cb

Compiler.**AinterB**(*la*, *lb*)
Returns the intersection between lists la and lb

Compiler.**sendMessAbsorp**(*bn*, *jt*, *ca*, *cb*)
Updates the compiler array with inscrutions to send the message (absorption) from ca to cb

Compiler.**collectAroundCliq**(*bn*, *jt*, *ca*, *index*, *diffu*)
Updates the compiler array to collect informations arround the cliq ca. index is the index of ca in diffu and diffutmp is the list of the first element of all elements of diffu until index

Compiler.**sendMessDiffu**(*bn*, *jt*, *ca*, *cb*, *index*, *diffu*)
Updates the compiler array with instructions to send the message (diffusion) from ca to cb

Compiler.**inference**(*bn*, *jt*, *absorp*, *diffu*, *targets*, *targetmp*)
Considering the targets of a bn, inference does the absorption and the diffusion of the information

Compiler.**output**(*bn*, *jt*, *target*, *diffu*)
Instructions for the last cliques to be normalized and return the results for respective targets

Compiler.**compil**(*bn*, *targets*, *evs*)
This function uses all the predefined functions above to fill the compiler array with instructions to get the targets of a bn according to evidences

Created on Wed Apr 22 21:10:26 2015

@author: Marvin