

Core to Lotos Translation Programmer Guide

Translation steps:

1. Parsing core text program file and getting xml version of core program:
2. reading the core xml file to memory (building Document):
 - In this step we validate the core xml input file ,so if the core xml file is not valid the translation will fail (error message shown)
 - notice that the parsing is done with SaxBuilder thus all the input file will be loaded into the memory.
 - The Input xml file must include the correct path to the cdl DTD.
 - In the case that the core xml not valid the program doesn't point to the error in the program so you may use xml spy to find the error.
3. Translating core xml file to Lotos xml file:

After we get the core xml file into the memory we start to build the lotos xml file in the memory , all the Global data needed for translation (e.g. input/output file name , the root of the core program , the global variables ...) , the steps of this translation are as appear in this semi-code:

```
Function main(){
    Global ← Save_global_data
    If( Global_data_exist) then
        Add_Global_Process
    Modules ← Get_all_modules.
    For-each module in Modules{
        Process ← Translate_module_to_process(module)
        Add_Process_to_translation_Tree(Process)
    }
}

Function Translate_module_to_process(module){
    If (theModuleHasTransitions){
        Transitions ← getAllTransitions(module);
        For-each transition in Transitions{
            Operation ← Translate_transition(transition);
            Add_Operation_to_current_process_Tree(Operation)
        }
    }
    else{ // we have modcombin
        modcombin ← getModCombin(module);
        translateModcombin(modcombin);
    }
}

Function translateModcombin(modcombin){
    If (ModcombIsInst(modcombin) ) translateInst(modcombin);
    If (ModcombIsSynch(modcombin) ) translateSynch (modcombin);
    If (ModcombIsASynch(modcombin)) translateASynch (modcombin);
    If (ModconIsPartSynch(modcombin) translatePartSynch (modcombin);
}
}
```

and so on ...for more detailed code see the source code.

- Notice that we add for the Lotos xml file and for the additional information file the relevant DTD for the with the correct path to the (the dtds are in the directory dtd) you can change the path by changing the config.xml file.
 - Also we add the xslt style-sheet for the Lotos xml file with the correct path to the xml2Lotos.xsl file , also you can change the path by changing the config.xml.
4. Transforming the xml file to a Lotos text file using XSLT (HTML text).
We transform the xml Lotos file to html version using xslt transformer.
 - You can use xml spy for transforming.
 5. Transforming the HTML file to text file using the program html2text (see Usage note while run).

How to change the path of the dtds:

There is a xml file named config.xml that contains:

```
<Config>
  <CDL_DTD>../dtd/cdl.dtd</CDL_DTD>
  <LOTOS_DTD>../dtd/lotos.dtd</LOTOS_DTD>
  <ADDINFO_DTD>../dtd/add_info.dtd</ADDINFO_DTD>
  <LOTOS_XSL>../xslt/xml2LOTOS.xsl</LOTOS_XSL>
</Config>
```

if you want you can change the path to the dtd file by changing the text value of the elements that represent the relevant dtd file for example to change the path of the `cdl.dtd` file to `z:\core2\lotos\cdl.dtd` change the config.xml to:

```
<Config>
  <CDL_DTD> z:\core2\lotos\cdl.dtd </CDL_DTD>
  <LOTOS_DTD>../dtd/lotos.dtd</LOTOS_DTD>
  <ADDINFO_DTD>../dtd/add_info.dtd</ADDINFO_DTD>
  <LOTOS_XSL>../xslt/xml2LOTOS.xsl</LOTOS_XSL>
</Config>
```

Comment: for more details see the file *core2lotos_implementation_design.doc*.