```
PROGRAM -> TASK_DEFINITIONS{PROGRAM.debts= TASK_DEFINITIONS.debts};
parbegin TASK_LIST parend

TASK_DEFINITIONS -> TASK_DEFINITION
{TASK_DEFINITIONS.debts=+TASK_DEFINITION.debts} TASK_DEFINITIONSTAG

TASK_DEFINITIONSTAG -> ;TASK_DEFINITION
{TASK_DEFINITIONSTAG.debts=+TASK_DEFINITION.debts}
TASK_DEFINITIONSTAG |epsilon

TASK_DEFINITION -> task id
{ if(task not already defined)TASK_DEFINITION.list.add(id,task)}
begin DECLARATIONS { COMMANDS } end

TASK_LIST -> task_id {TASKLIST.tasklist.add(id)}  TASK_LISTTAG

TASK_LISTTAG -> || task_id
{if(id not in tasklist) TASKLISTTAG.tasklist.add(id)}
TASK_LISTTAG|epsilon

DECLARATIONS -> DECLARATION  DECLARATIONSTAG

DECLARATIONSTAG -> ; DECLARATIONS|epsilon

DECLARATION -> integer  id  { if(id not already defined)
Decleration.id=id.id ,Decleration.type=integer
|  real id { if(id not already defined) Decleration.id=id.id
,Decleration.type=real}

COMMANDS -> COMMAND  COMMANDSTAG

COMMANDSTAG -> ; COMMAND COMMANDSTAG | epsilon

COMMAND -> id = EXPRESSION  | do COMMANDS until CONDITION od   |

                         send task_id . signal_id

{if(task is not defined) COMMANDS.debts.add(id,signal)} (PARAM_LIST)

 | accept signal_id {if(signal  is in debts list) delete from debts
COMMAND.tasklist.add(Command.id)} TASK (DECLARATIONS)  |

                              begin DECLARATIONS { COMMANDS } end

PARAM_LIST -> EXPRESSION PARAM_LISTTAG

PARAM_LISTTAG -> , PARAM_LIST | epsilon

EXPRESSION -> int_num |real_num | id {Expression.id=id.id)}
EXPRESSIONTAG

EXPRESSIONTAG -> binary_ar_op EXPRESSION{ExpressionTAG.id=id.id)}
|epsilon

CONDITION-> ( id rel_op )
```

Inherited:
```
Debts
```

Synthesized:
```
Command.id
Decleration.type
Decleration.id
ExpressionTAG.id
Expression.id
```