EDF R&D

INDUSTRIAL RISK MANAGEMENT

PROBABLISTIC SAFETY ASSESSMENT AND PROBABLISTIC GENERATION ASSESSMENT OF SYSTEMS

7 boulevard Gaspard Monge 91120 PALAISEAU - +33 (1) 78 19 32 00

06/22/2017

Syntax of the probabilistic modeling language Figaro

Marc BOUISSOU — EDF R&D - MRI

Lena BUFFONI — Linköping University

Jean-Christophe HOUDEBINE — Société Aristè

6125-1612-2017-01941-EN	1.0		
Information type : Technical note			

This document provides a detailed description of the Figaro language syntax. It is intended to accompany the language reference manual that explains the principles of modeling, the existence of two levels called order 1 and order 0, the semantics of the language and how to use it to build knowledge bases allowing the reliability modeling of discrete systems. This document is designed to be easy to use by a person familiar with the concepts of the Figaro language, but not the way to implement them. It presents successively the following aspects: 1) the language lexicon in English and French, 2) the language grammar, 3) grammatical controls performed on knowledge bases, 4) predefined functions of the language.

Circuit de validation

Auteur	Marc BOUISSOU	12/06/2017	M Bouissou
Vérificateur	Thomas CHAUDONNERET	13/06/2017	
Approbateur	Hélène PESME	22/06/2017	H. Pesme

Code affaire	**P11DD**

Liste de diffusion

Groupe destinataire
1612-EPSDS

Pré-diff	Diff	Destinataire	Structure	E-mail
	X	BOUISSOU Marc	EDF R&D - MRI	marc.bouissou@edf.fr
	X	BOUSKELA Daniel	EDF R&D - STEP	daniel.bouskela@edf.fr
	X	BUFFONI Lena	Linköping University	lena.buffoni@liu.se
	X	CHAUDONNERET Thomas	EDF R&D - MIRE	thomas.chaudonneret@edf.fr
	X	DONAT Roland	Société Edgemind	roland.donat@edgemind.net
	X	HOUDEBINE Jean-Christophe	Société Aristè	ariste@wanadoo.fr
	X	Jean-Christophe HOUDEBINE	Société Aristè	
	X	LAAROUCHI Youssef	EDF R&D - SINETICS	youssef.laarouchi@edf.fr
	X	Lena BUFFONI	Linköping University	
	X	PESME Helene	EDF R&D - MRI	helene.pesme@edf.fr
	X	THUY N	EDF R&D - STEP	n.thuy@edf.fr

AVERTISSEMENT / CAUTION

L'accès à ce document, ainsi que son utilisation, sont strictement limités aux personnes expressément habilitées par EDF.

EDF ne pourra être tenu responsable, au titre d'une action en responsabilité contractuelle, en responsabilité délictuelle ou de toute autre action, de tout dommage direct ou indirect, ou de quelque nature qu'il soit, ou de tout préjudice, notamment, de nature financière ou commerciale, résultant de l'utilisation d'une quelconque information contenue dans ce document.

Les données et informations contenues dans ce document sont fournies "en l'état" sans aucune garantie expresse ou tacite de quelque nature que ce soit.

Toute modification, reproduction, extraction d'éléments, réutilisation de tout ou partie de ce document sans autorisation préalable écrite d'EDF ainsi que toute diffusion externe à EDF du présent document ou des informations qu'il contient est strictement interdite sous peine de sanctions.

The access to this document and its use are strictly limited to the persons expressly authorised to do so by EDF.

EDF shall not be deemed liable as a consequence of any action, for any direct or indirect damage, including, among others, commercial or financial loss arising from the use of any information contained in this document.

This document and the information contained therein are provided "as are" without any warranty of any kind, either expressed or implied.

Any total or partial modification, reproduction, new use, distribution or extraction of elements of this document or its content, without the express and prior written consent of EDF is strictly forbidden. Failure to comply with the above provisions will expose to sanctions.

Contents

1. General information

This document provides a detailed description of the syntax of the Figaro language. **It is intended to come with the language reference manual** that explains the modeling principles, the existence of two levels called order 1 and order 0, the semantics of the language and how to use it to build knowledge bases enabling reliability-modeling of discrete systems.

This document is designed to be easy to use by a person familiar with the concepts of Figaro language, but not the details of implementing them; it presents, in succession:

- The lexical elements in English and in French
- The language grammar
- Grammatical controls on the knowledge base.
- Functions predefined in the language

Order 1 Figaro language allows describing generic knowledge, whereas order 0 Figaro can only be used to describe particular systems and serves as input to all models processing algorithms. In Figaro 1, there are two types of files: knowledge base files and system description files. Knowledge bases can contain only classes (CLASS) and permanent objects that are visible by all other objects (SYSTEM_OBJECT), while system descriptions may contain only objects (OBJECT). At order 0, there is only one type of file, which can contain only objects (OBJECT). The Figaro parser has two modes designed to check models at order 1 and order 0.

Apart a few details, that will be explicitly stated, the language at order 0 is a sub-language of order 1.

The writing conventions are as follows:

- “|”: indicates an exclusive choice between different options,
- [c]: indicates that the presence of the syntactic construction c is optional,
- *Italics*: indicates a keyword or expression *available only at order 1*.
- Parentheses are used to group elements: (a|b) c means (a or b) followed by c while a | (bc) means a or (b followed by c). The parentheses forming part of the Figaro text will be denoted LPAR and RPAR.

The distinction between upper and lower case is important. The terms TERM and term thus represent two different entities. All keywords of the Figaro language are capitalised.

2. Comments

Comments are allowed anywhere in the Figaro code. Any text between "(*" and "*)" is taken as a comment. Nested comments are allowed.

Example:

```
(* Comment
    (* Nested comment *)
*)
```

3. Lexical elements of the Figaro language

The lexical elements are the atomic terms of the language.

3.1. Construction of identifiers, numbers and delimiters

contenu	suite de caractères attendue
letter	A | B … | Z | a | b…| z | _
digit	0 | 1 ... | 9
charseq (character sequence)	(letter | digit) (| charseq)
identifier (called ID in syntactic diagrams)	letter charseq
character_string	sequence of arbitrary characters *except* " and |
LITERAL_STRING (string of characters between double quotes)	"character_string"
identifier between pipes (ID)	|character_string|
identifier between quotes (QUOTID)	’charseq’
integer (LITERAL_INTEGER)	digit (| integer)
real (LITERAL_REAL)	integer[.integer][(E|e)[-|+]integer]
end of Figaro statement	;
separator between identifiers, actions	,
opening and closing parenthesis (LPAR and RPAR in syntactic diagrams)	( )
designation prefix for a variable of an equations system	?

3.2. Keywords of the Figaro language

3.2.1. Main Figaro entities

These keywords are those that one will find at the highest level in the syntax tree of a Figaro text.

Figaro meaning	French term	English term
announcement of the list of steps	ORDRE_DES_ETAPES	STEPS_ORDER
announcement of a step condition	CONDITION	CONDITION
announcement of the list of groups	NOMS_DES_GROUPES	GROUP_NAMES
announcement of the knowledge base description	*DESCRIPTION_BDC*	*KB_DESCRIPTION*
announcement of the list of equations	NOMS_DES_SYSTEMES	SYSTEM_NAMES
announcement of a class	*TYPE*	*CLASS*
inheritance link of a class	*SORTE_DE*	*KIND_OF*

set of all objects (Figaro 1) announcement of an object (Figaro 1 and Figaro 0)	OBJET	OBJECT
announcement of a « permanent » and global object	*OBJET_SYSTEME*	*SYSTEM_OBJECT*
inheritance link of an object	EST_UN | EST_UNE	IS_A | IS_AN

3.2.2. Elementary operators

Figaro meaning	French term	English term
assignation	<--	<--
Boolean operator equality assignation in Figaro 0	=	=
Boolean operator different	<>	<>
Boolean operator strictly lower than	<	<
Boolean operator lower or equal	<=	<=
Boolean operator strictly greater than	>	>
Boolean operator greater or equal	>=	>=
multiplication	*	*
division	/	/
addition	+	+
subtraction	-	-
modulo (a % b means a modulo b)	%	%
power (a**b means a to the power b)	**	**
Boolean negation	NON	NOT
Boolean operator AND	ET	AND
Boolean operator OR	OU	OR

3.2.3. Interfaces

Figaro meaning	French term	English term
announcement of Figaro INTERFACES	INTERFACE	INTERFACE
announcement of the class of objects allowed in an interface	*GENRE*	*KIND*
announcement of the bounds of the number of objects allowed in an interface	*CARDINAL*	*CARDINAL*
announcement of the maximal bound of the number of objects allowed in an interface	*JUSQUA*	*TO*
infinite bound for the number of objects allowed in an interface	*INFINI*	*INFINITY*

3.2.4. Terms specific to occurrence rules

Figaro meaning	French term	English term
announcement of the occurrence rules	OCCURRENCE	OCCURRENCE
announcement of a rule condition that is additional to the implicit condition linked to the use of the keywords FAULT, UNAVAILABILITY or REPAIRS followed by the name of a FAILURE	SI	IF
announcement of a transition	IL_PEUT_SE_PRODUIRE	MAY_OCCUR
announcement of an alternative transition	OU_BIEN	OR_ELSE
standard transition	TRANSITION	TRANSITION
failure transition	DEFAILLANCE	FAULT
unavailability transition	INDISPONIBILITE	UNAVAILABILITY
repair transition	REPARATION	REPAIR
announcement of the list of failures repaired by the transition	*REPARE*	*REPAIRS*
announcement of the kind of probability distribution associated to the transition	LOI	DIST
announcement of the consequences of the transition	PROVOQUE	INDUCING
exponential probability distribution	EXP | EXPONENTIELLE	EXP | EXPONENTIAL
instantaneous probability distribution	INS | INSTANTANEE	INS | INSTANTANEOUS
fixed time “probability distribution”	T_C | TEMPS_CONSTANT	C_T | CONSTANT_TIME
other distributions, that can be used only in models to be processed by Monte Carlo simulation: cf. table in §6.	Same names as in English.	CYCLE, POINT, UNI (Uniform), TRIANG, ERL (Erlang), WEI | WEIBULL, GUMBEL, FRECHET, PARETO, NORMAL, LGN, GAMMA, BETA
version « with memory » of the preceding distributions. All distributions defining times are found here, except the exponential distribution which is memoryless by definition.	Same names as in English.	T_C_M, CYCLE_M, POINT_M, UNI_M, TRIANG_M, ERL_M, WEI_M , GUMBEL_M, FRECHET_M, PARETO_M, NORMAL_M, LGN_M, GAMMA_M, BETA_M

3.2.5. Terms specific to interaction rules

Figaro meaning	French term	English term
announcement of interaction rules	INTERACTION	INTERACTION
announcement of the steps the rule belongs to	ETAPE	STEP

announcement of the groups the rule belongs to	GROUPE	GROUP
announcement of the rule condition	SI	IF
announcement of the action executed when the condition is fulfilled	ALORS	THEN
announcement of the action executed when the condition is not fulfilled	SINON	ELSE

3.2.6. Common terms of interaction and occurrence rules

Figaro meaning	French term	English term
allows a rule to refer to the object that contains this rule	*MOI_MEME*	*MYSELF*
announces a variable that will represent an object taken in a set	*SOIT*	*GIVEN*
quantifier	*QQSOIT*	*FOR_ANY*
quantifier	*IL_EXISTE*	*IT_EXISTS*
announcement of a set of objects (interface or OBJECT)	*UN | UNE*	*A | AN*
announcement of the name of an object or interface with cardinal 1	DE	OF
announcement of a class name	*DE_TYPE*	*OF_TYPE*
announces a constant condition	*VERIFIANT*	*VERIFYING*
announcement of the condition associated to a quantifier FOR_ANY	*ON_A*	*WE_HAVE*
announcement of the condition associated to a quantifier IT_EXISTS	*TEL_QUE, TELLE_QUE*	*SUCH_THAT*
Boolean operator to test the set of FAILURE variables of an object	*MARCHE*	*WORKING*
Boolean operator to test the set of FAILURE variables of an object	*PANNE*	*FAILURE*
beginning of the expression at least K within(), or it exists at least K x included in … such that…	*AU_MOINS*	*AT_LEAST*
announcement of a list of Boolean expressions to be tested according to the mode at least K within()	*PARMI*	*WITHIN*
operator of testing the belonging of an object to a set (interface)	*INCLUS_DANS*	*INCLUDED_IN*
announces a set of objects on which an action is defined	*POUR_TOUT*	*FOR_ALL*
announces the actions associated with the quantifier FOR_ALL	*FAIRE*	*DO*
operator sum	*SOMME*	*SUM*

operator product	*PRODUIT*	*PRODUCT*
operator maximum	*MAXIMUM*	*MAXIMUM*
operator minimum	*MINIMUM*	*MINIMUM*
announces terms of operators sum, product, maximum and minimum	*DES_TERMES*	*OF_TERMS*

3.2.7. VARIABLES, EDITION, DOMAIN AND VALUES

Figaro meaning	French term	English term
announcement of a comment string of a slot of a class	LIBELLE	LABEL
announcement of Figaro constants	CONSTANTE	CONSTANT
announces probabilistic distribution parameters (considered as constants)	*PARAMETRE_LOI*	*DIST _PARAMETER*
announcement of Figaro attributes	ATTRIBUT	ATTRIBUTE
announcement of Figaro effects	*EFFET*	*EFFECT*
announcement of Figaro failures	*PANNE*	*FAILURE*
announcement of a characteristic's (constant, attribute, effect, failure) edition property	*EDITION*	*EDITION*
announcement of a variable's edition property	*VISIBLE*	*VISIBLE*
announcement of a variable's edition property	*MODIFIABLE*	*MODIFIABLE*
announcement of a variable's edition property	*OBLIGATOIRE*	*MANDATORY*
announcement of constraints on the uses of a characteristic. These constraints are related to the deduced objects construction algorithms. Cf. the different possible roles in the following bullets:	*ROLE*	*ROLE*
• no constraint on the use of a characteristic. Used to remove inherited constraints.	*STANDARD*	*STANDARD*
• constraint on the use of a CONSTANT: can not have its value set in a variant. Thus we can rely on this constant in the algorithms calculating deduced objects. By extension, it is a constraint that must be added to the parameters involved in the reliability models associated to failures, because the fault tree generator can not handle variants relating to the values of these parameters.	*CONCEPTION*	*DESIGN*
• indicates an interface used to determine whether a deduced object can be kept (in	*REFERENCE*	*REFERENCE*

the event that it has changed characteristics) instead of rebuilding it.		
• indicates a characteristic that is used only in the construction of the deduced objects, and has no use in the final Figaro model.	*PROVISOIRE*	*TEMPORARY*
announcement of a string associated with an object	DESCRIPTION	DESCRIPTION
announces the domain declaration of a variable	*DOMAINE*	*DOMAIN*
announces the expression defining the default initial value of a variable	*PAR_DEFAUT*	*DEFAULT*
announces the expression defining the reinitialisation value of a variable	REINITIALISATION	REINITIALISATION
	VRAI	TRUE
	FAUX	FALSE
	ENTIER	INTEGER
	REEL	REAL
	BOOLEEN	BOOLEAN
variable giving the current time (usable only in Monte Carlo simulation)	DATE_COURANTE	CURRENT_DATE
announcement of reliability models associated with a FAILURE	*DONNEES_FIABILISTES*	*RELIABILITY_DATA*
announcement of the default reliability model	*MODELE_PAR_DEFAUT*	*DEFAULT_MODEL*
announcement of the replaced reliability model	*MODELE_REMPLACE*	*MODEL_REPLACED*
reliability model: frequency	*MODELE_F*	*MODEL_F*
reliability model: constant value TRUE or FALSE	*MODELE_FIGE*	*MODEL_FROZEN*
reliability model: time-independent probability	*MODELE_G*	*MODEL_G*
reliability model: repairable, failures on demand and in operation	*MODELE_GLM*	*MODEL_GLM*
reliability model: failure on demand and in operation, fixed mission time	*MODELE_GLTM*	*MODEL_GLTM*
reliability model: randomly tested	*MODELE_TA*	*MODEL_RT*
reliability model: tested according to a fixed schedule	*MODELE_TPE*	*MODEL_PET*
reliability model: not repairable, Weibull law for time to failure	*MODELE_WB*	*MODEL_WB*
parameter name of a reliability model associated with a FAILURE	*ALPHA*	*ALPHA*
idem	*FREQUENCE*	*FREQUENCE*

idem	*GAMMA*	*GAMMA*
idem	*GAMMA_NON_DETECTION*	*GAMMA_NO_DETECTION*
idem	*GAMMA_RECONFIG*	*GAMMA_RECONFIG*
idem	*GAMMA_TEST*	*GAMMA_TEST*
idem	*LAMBDA*	*LAMBDA*
idem	*LAMBDA_TEST*	*LAMBDA_TEST*
idem	*MU*	*MU*
idem	*T_C_M*	*C_T_M*
idem	*T_INIT_TEST*	*T_INIT_TEST*
idem	*T_INTER_TEST*	*T_INTER_TEST*
idem	*T_TEST*	*T_TEST*
idem	*T0*	*T0*
idem	*TM*	*MT*

3.2.8. Linear equations systems

Figaro meaning	French term	English term
announcement of a list of equations	EQUATION	EQUATION
type of equations	LINEAIRE	LINEAR
announcement of the systems an equation belongs to	SYSTEME_EQUATIONS	EQUATION_SYSTEM
request for resolution of a system of equations	RESOUDRE_SYSTEME	SOLVE_SYSTEM
announcement of an equation	FORMULE	FORMULA

4. Syntax of the Figaro language

This chapter details the syntax of the language, i.e. the decomposition of the parse tree into increasingly elementary structures until arriving at the lexical elements described in the previous chapter.

Important note: since the last version of Figaro 1 the use of GLOBAL variables has been removed because it was a source of bugs; it is replaced by system objects.

NB: in order to increase the maintainability of this document, the "syntax diagrams" in this chapter are given only with the English vocabulary and keywords. Figaro keywords are in bold for easy identification and to differentiate them from terminal elements of the syntax which are also noted in capital letters (eg LITERAL_STRING). *What is in italics is only for order 1.*

The syntactic tree root of a Figaro text is Figaro_model.

Figaro_model:

(***KB_DESCRIPTION*** *LITERAL_STRING ;* | steps | groups | systems | class | object) [Figaro_model]

Constraints:

- There may be at most one description introduced by KB_DESCRIPTION in a knowledge base.
- The structures steps, groups, systems must be declared at most once.

4.1. Declaration of the steps list

steps: **STEPS_ORDER** liststeps

liststeps: list_ids [**CONDITION** term [label_string]]
| **LABEL** label_string **CONDITION** term
| [**LABEL** label_string] [liststeps] ;
| **STEP** LITERAL_INTEGER list_ids [liststeps] ;

list_ids: ID [, list_ids]

label_string: LITERAL_STRING

Constraints:

- A given step may appear only once in the list.

Remark:

- An interaction rule which is not explicitly associated to a step is put in the (always existent) step called default_step (etape_par_defaut in French). To control the rank of the default_step, the user must declare it in the steps list.

4.2. Declaration of the groups list

groups: **GROUP_NAMES** group_list

group_list: ID [**LABEL** label_string] ; [group_list]

Constraints:

- A given group may appear only once in the list.

4.3. Declaration of the equations systems list

```
systems: SYSTEM_NAMES system_list
system_list: ID [LABEL label_string] ; [system_list]
```

Constraints:

- A given equation system may appear only once in the list.

4.4. Definition of a class

```
class:
 CLASS ID  [GR_LINK|GR_NODE] [KIND_OF enum_ids] ; fields

fields:
    (const | attribute | effect | failure | dist_param | interface |
occurrence | interaction | equation | description) [fields]
```

Constraints:

- The GR_LINK and GR_NODE keywords are used only to indicate to a graphical editor (during its parameterisation) that objects of this class must be represented as a link or an icon. They should not normally remain in a finalised knowledge base.
- A class must be declared only once.
- Each parent class must appear only once in the parent list announced by KIND_OF.

4.5. Definition of an object

```
objet: (OBJECT | SYSTEM_OBJECT)
            ID IS_A ID ; objfields
objfields: (description | const | attribute | effect
      | failure | dist_param | interface | occurrence
      | interaction | equation)
description: DESCRIPTION LITERAL_STRING ;
```

Constraints:

- An object must be declared only once. It can not be declared in a knowledge base file.
- A system object is a special object, created immediately at initialisation and visible to all other objects in the system. It can therefore be used to share variable values. Such an object can only be declared in a knowledge base file.

4.6. Definition of an object or class characteristics

Hereafter, some structures shared by several characteristics.

```
label: LABEL LITERAL_STRING
```

```
domain: DOMAIN (INTEGER | REAL | BOOLEAN | enum_quote_ids)
enum_quote_ids: QUOTID [enum_quote_ids]
default_term: DEFAULT term
affect_term:  = term     (only for objects)
reinit_term: REINITIALISATION term
```

```
edition: EDITION terms_ed
terms_ed: (term_ed | NON term_ed
      | terms_ed , term_ed | terms_ed , NON term_ed)
terme_ed: (VISIBLE | MODIFIABLE | MANDATORY)
```

```
role: ROLE (STANDARD | DESIGN | REFERENCE | TEMPORARY)
```

A role is used to define constraints on the use of a characteristic. Cf. table of section 3.2.7 for more details.

Constraints:

- The domain can be set only once for a variable.
- An enumerated domain cannot contain the same value twice.
- The default value and the reinitialisation value (if any) of a variable are the same and can only be set once for each variable.
- The label can be set only once for a characteristic.

4.6.1. Characteristics definition

Constraints:

- A characteristic (CONSTANT, EFFECT, FAILURE, ATTRIBUTE, DIST_PARAMETER) can only be defined once in each class description.
- A characteristic cannot have the same name as an interface.
- For each characteristic, a number of "slots" have a name in the syntax diagrams beginning with f_. Each of these slots must be defined at most once.
- In a class, a characteristic can only be given a default (default_term) or a reinitialisation (reinit_term), value, whereas in an object it can only be given an initial (initial_term) or a reinitialisation (reinit_term) value.

4.6.1.1. Attributes

```
attribute: ATTRIBUTE attributes
attributes: (ID f_attribute | error) ; [attributes]
```

```
f_attribute: (edition | role | domain | default_term
      | reinit_term | affect_term | label) [f_attribute]
```

NB: the "structure" error (which will be seen in multiple places) can be ignored, because it is used by the parser to be able to continue the syntactic analysis even if an error is encountered in the structure being analysed, delimited by a " ; ". Thus the user can get multiple error diagnostics in a single pass of the syntax checker.

4.6.1.2. Constants

```
const: CONSTANT constants

constants: ID f_constant ; [constants]

f_constant: (edition | role | domain | default_term | affect_term | label)
[f_constant]
```

4.6.1.3. Probability distribution parameters

There is no substantive difference between the declaration of a DIST_PARAMETER and a *real* CONSTANT. There are two different keywords only to facilitate understanding of a knowledge base. A DIST_PARAMETER must be used only in the definition of probability distributions, as its name suggests, and cannot be used in Figaro 0.

```
dist_param: DIST_PARAMETER dist_params

dist_params: (ID f_dist_param ; | error) [dist_params]

f_dist_param: (edition | affect_term | role | default_term | label)
[f_dist_param]
```

4.6.1.4. Failures

```
failure: FAILURE failures

failures: ID f_failure ; [failures]

f_failure: (edition | affect_term | role | default_term | label | rel_data)
[f_failure]
```

4.6.1.5. Effects

```
effect: EFFECT effects

effects: (ID f_effect| error) ;  [effects]

f_effect: (edition | role | label) [f_effect]
```

4.6.2. Interfaces definition

```
interface: INTERFACE interfaces

interfaces: (ID f_interface | error) ; [interfaces]

f_interface: (edition | kind | cardinal | affect_ids | label | role)
[f_interface]

kind: KIND ID

cardinal: CARDINAL LITERAL_INTEGER
      [TO (LITERAL_INTEGER | INFINITY)]
```

```
affect_ids: = enum_ids

enum_ids: ID [enum_ids]
```

Constraints:

- A variable can not have the same name as an interface.
- The cardinal minimum must be less than or equal to the maximum.

4.6.3. Rules definition

Remark: for both types of rules (occurrence and interaction), we find the following structures:

```
condition: IF term

group: GROUP enum_ids
```

Constraint: these two structures must appear only once in a rule.

4.6.3.1. Occurrence rules

```
occurrence: OCCURRENCE occurrences

occurrences: ([ID] f_occurrence MAY_OCCUR

      alternatives | error) ; [occurrences]

f_occurrence: (group | condition | label) [f_occurrence]

alternatives: [alternatives OR_ELSE] alternative

alternative: (TRANSITION ID f_transition

      | FAILURE ID f_failure

      | REPAIR ID f_reparation

      | UNAVAILABILITY ID f_unavail)


f_transition: (label | dist | induces) [f_failure]

f_failure: (label | dist | induces) [f_failure]

f_unavail: (label | dist | induces) [f_unavail]

f_repair: (label | dist | repairs | induces) [f_repair]

dist: DIST ID LPAR term RPAR

induces: INDUCING action

repairs: REPAIRS enum_ids
```

Constraints:

- An occurrence rule (identified by its ID) can be set only once per type.
- An occurrence rule must have at least one transition. An occurrence rule containing more than one alternative (hence containing the keyword OR_ELSE) must contain only INSTANTANEOUS type probability distributions.
- All transitions in the same rule must have different names.
- The DIST slot and the INDUCES slot must not be defined more than once for each transition.
- The REPAIRS slot must not contain the same repair twice.

- The GROUP slot must contain at least one group defined in the group list.

4.6.3.2. Interaction rules

```
interaction: INTERACTION interactions
interactions: ([ID] f_interaction | error)  ; [interactions]
f_interaction: (step | group | condition | label | given | then | else)
[f_interaction]
step: STEP enum_ids
given: (GIVEN anobject | given AND anobject | given , GIVEN anobject)
then: THEN action
else: ELSE action
anobject: ID (A|AN) obj  [OF_TYPE ID] [VERIFYING term]
```

Constraints:

- An interaction rule (identified by its ID) can be set only once per type.
- An interaction rule must have at least the THEN slot.
- The STEP slot must contain at least one step defined in the step list.
- The GROUP slot must contain at least one group defined in the group list.

4.6.4. Equations

```
equation: EQUATION equations
equations: [ID f_equation | error] ; [equations]
f_equation: (kind_eq | system_eq | condition
      | formula_eq | label) [f_equation]
kind _eq: KIND LINEAR
systeme_eq: EQUATION_SYSTEM ID
formula_eq: FORMULA term = term
```

Constraints:

- An equation (identified by its ID) can be defined only once per type.
- The EQUATION_SYSTEM slot must use the name of a system defined in the system list.

4.6.5. Actions

```
action: (action_elem
      | action , action | LPAR action RPAR)
action_elem: assign
      | FOR_ALL anobject DO action
      | SOLVE_SYSTEM LPAR ID RPAR)
assign: (var <-- term | var | NOT var)
```

Constraints:

- The system used in the slot SOLVE_SYSTEM must exist in the system list.

4.6.6. Definition of expression terms

```
term: constant | var
        | term (> | >= | = | <> | < | <=) term
        | isworking | failed
        | term (AND | OR) term | NOT term
        | FOR_ANY anobject WE_HAVE term
        | IT_EXISTS anobject SUCH_THAT term
        | IT_EXISTS AT_LEAST term anobjectin SUCH_THAT term
        | AT_LEAST term WITHIN LPAR list_sep_terms RPAR
        | obj INCLUDED_IN obj
        | CARDINAL OF obj
        | CARDINAL LPAR obj RPAR
        | term( + | - | * | / | ** | % ) term
        | term ?= LPAR free_terms RPAR
        | LPAR error RPAR
        | LPAR term RPAR
        | - term
        | ?var             (warning: no blank character after "?")
        | SUM FOR_ALL anobject OF_TERMS term
        | PRODUCT FOR_ALL anobject OF_TERMS term
        | MAXIMUM FOR_ALL anobject OF_TERMS term
        | MINIMUM FOR_ALL anobject OF_TERMS term
        | RAND
        | CURRENT_DATE
        | INTEGRAL LPAR term RPAR
        | ALREADY_REALIZED LPAR term RPAR
        | STATE_TIME LPAR term RPAR

anobjectin: ID INCLUDED_IN obj [OF_TYPE ID] [VERIFYING term]
isworking: WORKING [LPAR obj RPAR | OF obj]
failed: FAILURE [LPAR obj RPAR | OF obj]
var: ID [LPAR var RPAR | LPAR list_terms RPAR | OF var]
list_terms: term [, list_terms]
list_sep_terms: term [, list_sep_terms]
constant: (LITERAL_INTEGER | LITERAL_REAL | TRUE | FALSE | QUOTID)
obj: ID (LPAR obj RPAR | OF obj) | MYSELF
```

Constraints:

- The keywords RAND, CURRENT_DATE, INTEGRAL, ALREADY_REALIZED and STATE_TIME can be used only in models intended to be exploited by Monte Carlo simulation.

4.6.7. Definition of a reliability model and related data

This structure is used to define one or more leaf models associated with a FAILURE in order to generate fault trees.

Reliability models can only be declared in Figaro 1.

Constraints:

- Several models can be defined, but only one of them can be declared as the default model.

```
rel_data: RELIABILITY_DATA rel_elements

rel_elements:
     | rel_elements condition
     | rel_elements group
     | rel_elements DEFAULT_MODEL model_type
     | rel_elements replaces_model
     | rel_elements rel_model
     | rel_elements error


replaces_model: MODEL_REPLACED | REPLACES term


rel_model: model_type rel_parameters


rel_parameters: | rel_parameter [rel_parameters]


rel_parameter: ID = term


model_type:
     MODEL_GLM | MODEL_FROZEN | MODEL_GLTM | MODEL_G | MODEL_F | MODEL_RT |
     MODEL_PET | MODEL_WB
```

Reminder of corresponding French names:

```
     MODELE_GLM | MODELE_FIGE | MODELE_GLTM | MODELE_G| MODELE_F  |
     MODELE_TA | MODELE_TPE    | MODELE_WB
```

5. Grammatical constraints of Figaro 1 language

In the preceding chapters, constraints have already been stated; they were simple because local to the syntactic structures described. Respecting these constraints is not sufficient to ensure the overall consistency of a Figaro model.

The purpose of this chapter is to describe the more global constraints that also exist. The respect of these constraints is checked by the Figaro tools. Failure to comply with these instructions will result in warnings or errors.

5.1. Grammatical constraints related to inheritance

Inheritance-related grammatical constraints first require the existence of at least one class in the knowledge base.

A class is described by its static characteristics (constants, variables and interfaces) and dynamics (occurrence and interaction rules, equations). Each characteristic of the parent classes of a class is copied at the level of the class in order to implement the overriding of the slots.

The overriding mechanism is implemented through the following two principles:

- Inheritance of the characteristics is done in the order of declaration after the keyword KIND_OF, by names (if two variables have the same name, there is overriding) and by slots (only the overridden slots are modified, the others are preserved).
- "*The last speaker is right*": in case of multiple inheritance, it is the order of the names specified in the slot KIND_OF which determines the order of taking into account the characteristics and, by the way, that of their facets. Thus, in the following example:

```
CLASS A ;

ATTRIBUTE weight DOMAIN REAL DEFAULT 14 ;

CLASS B ;

ATTRIBUTE weight DOMAIN REAL DEFAULT 18 ;

CLASS C KIND_OF B A ;
```

  The default value of `weight` in C is 14.

Inheritance resolution requires the following grammatical constraints on name and slot overriding and on the inheritance graph between classes:

- Name overriding:
  - ◊ The same name can not be used for a characteristic in a parent class X and for a characteristic of a different type in a class inheriting from X.
  - ◊ A class can not inherit two characteristics having the same name but different characteristic types, coming from two different classes.
  - ◊ It is recommended to use different names for an interaction rule defined in a parent class and an occurrence rule defined in a class inheriting from the parent class, and vice versa.
  - ◊ It is recommended not to use the name of an inherited transition to define a new transition in an inheriting class.
- Slot overriding:
  - ◊ The domains of the inherited characteristics must not be redefined in the inheriting classes except for the enumerated domains.
  - ◊ The KIND declared for an INTERFACE must be a subclass of the class declared as KIND in the interface of the same name of the parent class.
- It is forbidden to create a loop in the inheritance links.

5.2. Grammatical constraints related to interfaces

The grammatical constraints related to the interfaces are as follows:

- The class referenced by the KIND slot of an interface must be defined in the knowledge base.
- It is recommended that you do not use the same name for an interface and a class.

5.3. Grammatical constraints related to expressions

The grammatical constraints related to expressions are as follows:

- For a characteristic:
  - ◊ The default values of constants CONSTANT and DIST_PARAMETER must be constant expressions, that is, numerical or literal values involving only other constants. There should be no circular definition between constants.
  - ◊ The expressions of the default values must be compatible with the characteristic domains. The table below shows the authorised combinations.
  - ◊ Reinitialisation values of ATTRIBUTE variables must be constant expressions, that is, numerical or literal values involving only constants.
  - ◊ The expressions of the reinitialisation values must be compatible with the variable domains. The table below shows the authorised combinations.
  - ◊ The FAILURE and WORKING operators must be applied to objects with FAILURE variables.
  - ◊ The following table shows the casting rules applied when setting default values or reinitialisation values.

Variable domain	Expression domain	Authorised
BOOLEAN	BOOLEAN	Yes
BOOLEAN	INTEGER	No
BOOLEAN	REAL	No
BOOLEAN	Enumerated	No
INTEGER	BOOLEAN	Yes (taken equal to 0 or 1)
INTEGER	INTEGER	Yes
INTEGER	REAL	No
INTEGER	Enumerated	No
REAL	BOOLEAN	Yes (taken equal to 0 or 1)
REAL	INTEGER	Yes
REAL	REAL	Yes
REAL	Enumerated	No
Enumerated	BOOLEAN	No
Enumerated	INTEGER	No
Enumerated	REAL	No
Enumerated	Enumerated	*Yes, if and only if a value is common to both domains*

- More generally, within an expression, a Boolean is cast in integer (0 or 1) when it is one of the operands of a numeric operation whose other operand is a number. The result of an operation between an integer and a real is a real.
- For an equation:
  - The unknowns of an equation must be real attributes.
  - An equation has a type that can only be LINEAR:
    - The * operator must have only one of its operands containing an unknown, the other must not contain any.
    - The / operator must not contain unknowns in its right operand.
    - The PRODUCT operator has two operands, the first declares a linked variable and the second is an expression. This last expression must contain no unknown unless the cardinal of the set on which the linked variable is defined is 0 or 1.
    - The SUM operator has two operands, the first declares a linked variable and the second is an expression. This last expression must respect the rules announced on the operators * and /.
  - The IF slot used to describe the condition for applying an equation must be a Boolean expression.
- For a rule in general:
  - If at least one rule in the knowledge base belongs to a group, it is recommended that all rules in the database belong to at least one group.
- For an interaction rule:
  - If at least one interaction rule belongs to a step, it is recommended that all interaction rules belong to a step.
- For an occurrence rule:
  - Only the last transition of an instantaneous rule can be dispensed from the DIST slot. In all other cases, the DIST slot is mandatory.
  - Transitions corresponding to different types of distributions can not belong to the same occurrence rule.
  - The number of transitions defined in an occurrence rule must be at least one for the instantaneous rules and equal to one for all other rules.
  - For actions associated to a transition (a transition without explicit or implicit action is an error):
    - If the transition is neither a failure, nor a repair, nor an unavailability, the INDUCING slot is mandatory.
    - If the transition is a failure and its name does not match a FAILURE variable, the INDUCING slot is mandatory.
    - If the transition is a repair:
      - If the REPAIRS slot exists, it must be applied to (at least) a FAILURE variable.
      - If the REPAIRS slot is not specified, the slot INDUCING is mandatory.
    - If the transition is an unavailability,
      - If the name of the transition does not match a FAILURE variable, the INDUCING slot is mandatory.
      - The transition must be instantaneous.
- For a general expression:
  - The VERIFYING slot must be an expression that invokes only DESIGN constants.

◊ The two operands of a binary operator must have compatible domains. The following table gives for various operators the domain of the expression, the number and the type of the expected operands:

Operator	Expression domain	First operand	Second operand
OF_TYPE	Object	Selection expression	Class name
VERIFYING	Object	Selection expression	Boolean expression
AND	Boolean	Boolean expression	Boolean Expression
OR	Boolean	Boolean expression	Boolean expression
NOT	Boolean	Boolean expression	
INCLUDED_IN	Boolean	Expression defining a set of objects with cardinal 1	Expression defining a set of objects
CARDINAL	Entier	Expression defining a set of objects	
FAILURE	Boolean	Expression defining a set of objects with cardinal 1	
WORKING	Boolean	Expression defining a set of objects with cardinal 1	
FOR_ANY… WE_HAVE…	Boolean	Expression defining an alias	Boolean expression
IT_EXISTS… SUCH_THAT…	Boolean	Expression defining an alias	Boolean expression

6. Appendix: List of Functions

The application of a function named ID to a list of arguments is represented in the grammar by ID then "(" list of terms separated by commas ")". This syntax is generally accepted in order to avoid changing the Lex and Yacc files each time a function is added. When it encounters this type of construction, the Figaro parser looks for the function in a list.

The functions have the same name in English and French, except for the special functions listed in § 6.3.

6.1. Mathematical Functions

In the table below, n denotes an integer, r denotes a real number and x, y and z can be integers or real numbers.

Name of the function	Syntax
Exponential	EXP(r)
Sine	SIN(r)
Cosine	COS(r)
Tangent	TAN(r)
Square root	SQRT(r)
Neperian Logarithm	LN(r)
Logarithm in base 10	LOG(r)
Absolute value	ABS(r)
Closest larger integer	CEIL(r)
Closest smaller integer	FLOOR(r)
Closest integer	NINT(r)
Factorial (n!)	FACT(n)
Logarithm in base 10 of factorial (log(n!))	LFACT(n)
Maximum	MAX(x, y, z,…)
Minimum	MIN(x, y, z,…)

6.2. Functions returning a random number according to a given probability distribution

All the functions mentioned in this section can only be used in a Monte Carlo simulation.

RAND_EXP, RAND_POINT, RAND_UNI, RAND_TRIANG, RAND_ERL, RAND_WEI, RAND_GUMBEL, RAND_FRECHET, RAND_PARETO, RAND_NORMAL, RAND_LGN, RAND_GAMMA, RAND_BETA.

The definition of the probability distributions and their parameters are the same as those found in the definition of random firing times for timed transitions. The table below gives these definitions.

Remark: we do not find RAND_CYCLE in the list above, because CYCLE has a very specific meaning. This key word is used as a "distribution" associated with a transition to model the fact that this transition is triggered at regular intervals, in a deterministic way.

Name	Number of arguments	Type of arguments	Comment
EXP(λ)	1	real	Exponential distribution λ is the occurrence rate
CYCLE(T,T0)	2	real	"Distribution" for a cyclic event T is the period of the cycle T0 is the instant of the first event
POINT(T1,F1,…,Tn,Fn)	2*n	real	Distribution defined point by point by its cumulative distribution function 0<=Tn<=Tn+1 0<=Fn<=Fn+1<=1
UNI(A,B)	2	real	Uniform on [A,B] (A<=B)
TRIANG(A,M,B)	3	real	Triangular on [A,B] with mode at M (A<=M<=B)
ERL(λ1,K1,…,λn,Kn)	2*n	Odd arguments: real, Even arguments: integer	Generalised Erlang distribution
WEI(S,T,A,β)	4	first argument: integer, other arguments: real	Weibull law If S = 1, A is the mean, If S = 2, A is the median, If S = 3, A is the scale. T = offset (T>=0), β = shape parameter (β>0)
GUMBEL(m,σ)	2	real	Gumbel law m is the mean (m>0) σ is the standard deviation (s>0)
FRECHET(T,α,β)	3	real	Fréchet law T = offset (T>=0)

Name	Number of arguments	Type of arguments	Comment

			α = scale (α>0)
			β = shape parameter (β>0)

PARETO(T,σ,β,δ)	4	real	Pareto law
			T = offset (T>=0)
			σ = scale (σ>0)
			β = shape parameter (β>0)
			δ = inverse shape parameter (δ>0)

NORMAL(m,σ)	2	real	Normal distribution
			m = mean
			σ = standard deviation (σ>0)

LGN(S,A,B)	3	first argument: integer, other arguments: real	Log normal distribution If S = 1, A = mean and B = std deviation If S = 2, A = median and B = error factor If S = 3, A = mean and B = error factor
GAMMA(S,T,A,B)	4	first argument: integer, other arguments: real	Gamma distribution If S = 1, A = mean and B = std deviation If S = 2, A = scale and B = shape param.
BETA(p,q,A,B)	4	real	Beta distribution with parameters p,q on interval [A, B] p>0, q>0, A<B

6.3. Special functions

All the functions mentioned in this section can only be used in a Monte Carlo simulation.

In the table below, b denotes a Boolean expression and x a real, integer or Boolean expression.

Figaro meaning	French term	English term
Boolean variable indicating that argument b passed to TRUE at CURRENT_DATE or earlier	DEJA_REALISE(b)	ALREADY_REALIZED(b)
Calculation of the integral from 0 to CURRENT_DATE of the expression x in argument.	INTEGRALE(x)	INTEGRAL(x)
Cumulated time spent in the state defined by the indicator expression b, between 0 and CURRENT_DATE.	TEMPS_SEJOUR(b)	STATE_TIME(b)