Langage de production : description cohérente du court terme (première partie)

Jean Hilger

INRIA

UNITE DE RECHERCHE
INRIA-LORRAINE

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 912

LANGAGE DE PRODUCTION : DESCRIPTION COHERENTE DU COURT TERME (PREMIERE PARTIE)

Programme 5

Jean HILGER

Octobre 1988

RR_0912

Langage de Production:

description cohérente du court terme

(PREMIERE PARTIE)

Production Language:
coherent description of the short term

(FIRST PART)

Jean HILGER

Projet SAGEP

INRIA-LORRAINE
Technopole de NANCY-BRABOIS
Campus Scientifique
Bd des Aiguillettes, Bp 239
54506 Vandoeuvre-Les-Nancy CEDEX

* Ce travail a été partiellement financé par le Groupement Scientifique Mathématiques appliquées, Informatique et Gestion associant la FNEGE, INRIA et CNRS (groupe de travail LAPON).

Abstract

In this paper we present a first approach to a description and command language for hierarchical management of discrete production systems. The language is designed to guarantee coherency of any description of a hierarchical production management system (HPMS).

Looking at the short term level of production system management as a information system we aim to list all important data in order to represent their predominant links by an entity-relationship model. We design a language which permits to express non redundantly all relations between entities and we give the context free grammar which generates the sentences of our language. Finally we add an implementation of a lexical and syntactical analyser.

KEY-WORDS: PRODUCTION MANAGEMENT, LANGUAGE, INFORMATION SYSTEM, ENTITY - RELATIONSHIP MODEL, GRAMMAR

Résumé

Nous présentons dans ce papier une première approche d'un langage de description et de commande dédié à la gestion hiérarchisée de systèmes de production discrets. Nous développons un langage qui garantit la cohérence des descriptions qu'il permet de réaliser.

Notre but dans cette première partie de nos travaux est de répertorier les informations indispensables à la gestion de systèmes de production au niveau du court terme et de mettre en évidence les liens qu'ils entretiennent par la méthode du modèle entités - associations. Nous construisons un langage qui permet d'exprimer toutes les associations entre entités et nous donnons une grammaire context free qui permet de générer les phrases du langage. Nous fournissons également une implémentation d'un analyseur syntaxique et lexical pour notre langage.

MOTS-CLE: GESTION DE PRODUCTION, LANGAGE, SYSTEME D'INFORMATION, MODELE ENTITES - ASSOCIATIONS, GRAMMAIRE

INTRODUCTION

Objectif d'un langage de production

Nos recherches ont pour objectif la conception d'un outil de normalisation de la formulation des problèmes de gestion de systèmes discrets à l'aide d'un langage unifié de description et de commande que nous appelons par la suite langage de production. Ces recherches ont été motivées par l'extrême complexité des systèmes de production, la variété des concepts proposés pour les structurer et la multiplicité des modèles établis pour les gérer.

Dans notre approche *le langage de production se définit comme un langage de description et de commande cohérent de la gestion hiérarchisée de la production.* Nous considérons le système de production comme un système d'information, c'est-à-dire un ensemble structuré d'informations concernant la gestion de la production.

Notre *langage est hiérarchisé* en référence à un modèle de gestion hiérarchisé de la production à plusieurs niveaux. La gestion hiérarchisée de la production permet de réduire la complexité des problèmes de gestion à résoudre en ce sens que chacun des niveaux hiérarchiques considérés ne traite que des informations qui lui sont spécifiques afin d'aboutir à une solution optimale. Cette solution optimale constitue les contraintes appliquées au niveau immédiatement inférieur, s'il existe. Sinon, cette solution optimale est l'ensemble des décisions à appliquer au système de production. De plus chaque niveau hiérarchique manipule des informations globales par rapport au niveau hiérarchique inférieur. On parle d'agrégation et de décomposition de l'information dans la structure verticale de la hiérarchie. Ainsi l'étude du système d'information d'une gestion hiérarchisée de la production amène d'abord à définir pour chaque niveau les types d'informations gérés et à déterminer leurs liens à ce niveau, mais aussi à spécifier les liens d'agrégation et de décompostion entre types d'informations de niveaux consécutifs.

Ce double besoin de spécification nécessite une méthode rigoureuse: la spécification algébrique de types abstraits que nous nous proposons d'exposer dans une prochaine publication. L'intérêt de cette méthode de spécification sera de fournir, à un niveau hiérarchique donné, un cadre de définition rigoureuse du système d'information dans lequel nous pourrons effectuer des descriptions et formuler des commandes ou décisions concernant les données de la gestion.

Notre langage est un *langage de description* en ce sens qu'il fait intervenir dans ses phrases les types d'informations intéressant un niveau hiérarchique donné et exprime les liens entre informations.

Notre langage est un *langage de commande* en tant qu'il permet de formuler des décisions résultant de processus variés (Recherche Opérationnelle, Systèmes Experts, etc ...).

Notre langage permet des *descriptions et commandes cohérentes* en ce sens qu'elles sont prouvées valides à partir de la spécification équationnelle des types d'informations.

Les champs d'application

Les *applications futures* d'un tel langage se situent à la fois dans le domaine de la recherche et dans le domaine industriel.

Le langage se veut un outil normalisé. Il n'engendrera pas nécessairement le consensus sur la nature des problèmes de gestion, mais permet une description compréhensible et commune des problèmes à formuler.

Le langage de production se destine, comme le Langage de Conception Structuré (LCS) en algorithmique, à servir de support de réflexion algorithmique pour la résolution de problèmes d'ordonnancement, de calcul de charges, de besoins (...) par quelques structures de contrôle (début ... fin, si ... alors... sinon, tantque ... faire ... refaire) , et, de plus, il verifiera la sémantique des algorithmes proposés par la garantie de cohérence qu'il apporte par rapport à un univers défini et spécifié.

Au plan pratique notre langage se veut un langage pour la rédaction de cahiers des charges d'un système de gestion nouveau. Il peut servir d'outil de description du système d'information d'un système de production particulier et ainsi conduire à l'élaboration du schéma conceptuel de données permettant l'implantation d'une base de données des données nécessaire à la gestion du système de production. Le langage peut aussi être une interface d'interrogation de base de données bien plus compréhensible et adaptée au domaine de la gestion de la production qu'un langage informatique tel que le langage de manipulation de base de données d'IDS II, DL/1 d'IMS ou SQL d'ORACLE.

Dans ce papier ...

... nous nous limitons à la définition du langage utilisé au niveau hiérarchique le plus bas: la gestion du court terme, ou niveau de l'ordonnancement. Ainsi notre langage s'articule autour des problèmes de gestion que sont l'ordonnancement des opérations, leur exécution et la constitution de gammes permettant de réaliser des produits, de la gestion des stocks et des moyens de transports. Dans un premier paragraphe nous définissons formellement un modèle entités - associations pour ensuite représenter l'analyse du système d'information du court terme par un schéma entités - associations. Cette étape nous permet de répertorier les types d'informations essentiels à la gestion du court terme et de représenter leurs liens. Dans un deuxième paragraphe nous donnons une grammaire qui, construite sur la définition formelle du modèle entités - associations, génère des phrases qui décrivent de façon unique les associations entre entités du schéma général. Nous écrivons la structure des phrases générées et nous commentons leur signification. Un exemple de description d'une gamme et de son exécution permet de montrer l'utilisation de ce langage. Cet exemple a également été vérifié par un analyseur lexical et syntaxique intégré dont le programme est donné en annexe. Un troisième paragraphe évalue le résultat de cette approche à l'égard des critères établis dans cette introduction. En conclusion, nous esquissons la suite des travaux qui nous permettent de réaliser les objectifs du langage de production.

I. Le modèle entités - associations

I.1. Définition du modèle

Le modèle entités - associations est largement connu dans le domaine de l'analyse de systèmes d'information et se présente sous de nombreuses variantes. Nous en donnons ici une définition formelle qui se réfère aux concepts du modèle de base dans [CHEN76]. Cette définition formelle nous servira à déterminer la structure d'un langage qui décrit les liens entre entités d'information établis dans un schéma entités - associations.

Nous définissons le modèle entités - associations par un quadruplet $(\mathfrak{D}, \mathfrak{E}, \mathfrak{R}, \mathfrak{A})$ où

- $\mathfrak{D}$ est un ensemble de domaines de base $D_l \in \mathfrak{D}$ qui décrivent des ensembles typés de valeurs: entiers, réels, caractères, chaînes de caractères, (...) ,

- $\mathfrak{E}$ est un ensemble d'entités $E_i \in \mathfrak{E}$. Une entité E_i décrit un ensemble d'objets ou occurrences d'entités qui possèdent des caractéristiques communes.

- $\mathfrak{R}$ est un ensemble d'associations $R_k \in \mathfrak{R}$. R_k est une associations entre entités $E_1, E_2, E_3, ..., E_n$. Elle décrit un ensemble d'occurrences d'associations reliant les objets $e_1 \in E_1, e_2 \in E_2, ..., e_n \in E_n$, i.e. $R_k = \{ [e_1, e_2, ..., e_n] / e_1 \in E_1, e_2 \in E_2, ..., e_n \in E_n \}$.

- $\mathfrak{A}$ est un ensemble de fonctions d'attributs. Un attribut $A_l \in \mathfrak{A}$ est une fonction qui décrit une caractéristique de l'entité

$A_l : E_i \rightarrow D_l$

ou d'une association d'entités

$A_l : R_k \rightarrow D_l$.

Les symboles utilisés sont:

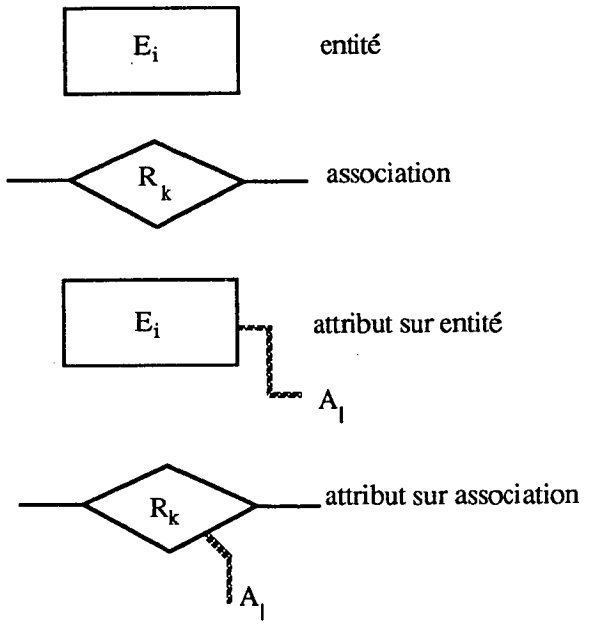


figure 1 : les symboles du modèle des entités - associations

Nous ne nous intéressons pas ici aux extensions données à ce modèle de base dans [HAWR84], [THEO86] et [DILE88], et, plus précisément aux possibilités de représentation de la notion de sous-ensemble ou de parties d'ensembles ainsi que l'ajout de contraintes sur les entités, les attributs et les associations. Notons simplement que l'écriture des contraintes dans les modèles étendus est orientée vers une implémentation du schéma de base de données ce qui n'est pas notre propos ici. En outre cette écriture ne permet pas de vérifier la non contradiction de l'ensemble des contraintes, ce qui est notre but.

I.2. Le schéma des données du court terme

Le système d'information à analyser est celui de la gestion au niveau bas de la hiérarchie dans le contexte d'une fabrication discontinue où nous rencontrons:

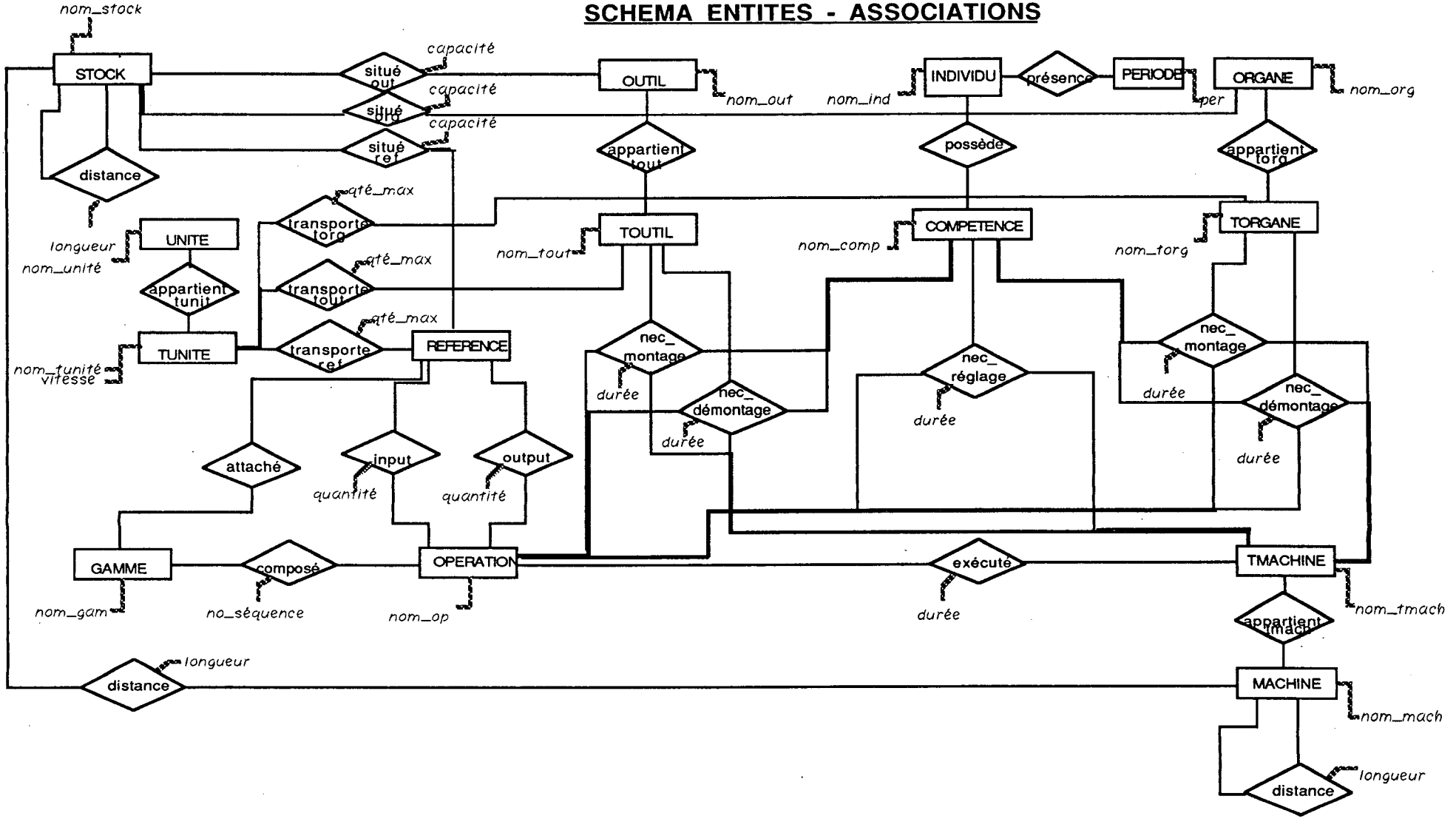
- des ressources (machines, moyens de transports et individus),
- des outils et organes d'assistance que l'on peut monter sur des machines pour exécuter des opérations différentes,
- des stockeurs à capacité limitée ou non,
- des durées opératoires, des durées de changement d'outils, d'organes d'assistance, de réglages de machines, de transport, de présence d'individus,
- des compétences nécessaires à la réalisation d'opérations,
- des distances entre machines, stocks, entre machines et stocks
- des références (produit fini, composant, en-cours ou matière première),
- des gammes multiples par référence de produit fini.

Nous considérons qu'une gamme est composée d'une suite ordonnée d'opérations qui elles-mêmes sont définies par un ensemble des références en entrée et en sortie de l'opération. Chaque gamme est rattachée à une référence de produit fini auquel elle conduit. Les opérations sont exécutées sur un type de machine auquel appartient un ensemble de machines physiques. L'exécution d'une opération peut nécessiter le montage, le démontage de types d'outils et types d'organes d'assistance et un réglage. Ces opérations supplémentaires mettent une durée déterminée et font appel à des compétences données. Les compétences sont les caractéristiques d'individus présents à des périodes déterminées. Les types d'outils et types d'organes d'assistance représentent respectivement des outils et des organes d'assistance physiques. Les stocks peuvent contenir un nombre limité ou non de références, d'outils et d'organes d'assistance. Les unités de transport sont regroupées par types d'unités de transport caractérisés par des vitesses de déplacement. Elles permettent de transporter une quantité maximale de références, d'outils et d'organes d'assistance sur des trajets entre positions physiques de ressources fixes.

Le schéma de données (figure 2) structure le système d'information du court terme. Nous donnons en annexe I la liste commentée des entités, des associations et de leurs attributs.

Le système d'information que nous présentons fait abstraction de tous les aspects non indispensables à la gestion du court terme. Par exemple, il n'est pas nécessaire de descendre au niveau des phases élémentaires qui se déroulent sur une même machine: le temps de séjour sur la machine suffit pour la recherche d'un ordonnancement; de même, on ne s'intéresse pas aux aspects morphodimensionnels des références à produire - qui ne sont connues que par leur nom -, ni aux problèmes de tolérance; autre exemple: le contrôle de la qualité n'intervient que par la durée de l'opération qu'il nécessite, mais les consignes qui définissent le processus de contrôle de qualité n'interviennent pas. Il est donc clair que le système d'information que nous proposons sera incomplet aux yeux du technicien de la production. Cependant, comme nous le verrons plus loin, l'univers décrit n'est pas clos, mais peut être enrichi par des types d'informations plus complets et des liens plus complexes.

SCHEMA ENTITES - ASSOCIATIONS



II. Un langage pour le modèle entités - associations

II.1. Définition du langage

Nous construisons un langage pour le modèle entités - associations défini en **I.1.**. Ce langage est l'ensemble de phrases qui expriment les occurrences d'associations entre entités représentées par les valeurs prises dans les domaines de leurs attributs.

Le langage L correspondant au modèle entités - associations est défini par un ensemble de phrases PH_k qui décrivent chaque association k de mk entités, pour k = 1..r, où r est le nombre d'associations du modèle et mk le nombre d'entités associées par une association k.

$$PH_k = \text{'['}\ \text{nom\_de}(R_k)\ \ {}^{A}/_{E1}\ \ (({}^{A}/_{E2}\ldots\ {}^{A}/_{Emk})\ (\varepsilon + \text{'''}\ {}^{A}/_{Rk}\ \text{'''}))^{+}\ \text{']'} \qquad (1)$$

$$L = \{\ PH_k / k = 1..r\ \}$$

où nom_de(R_k) est le nom de l'association R_k,

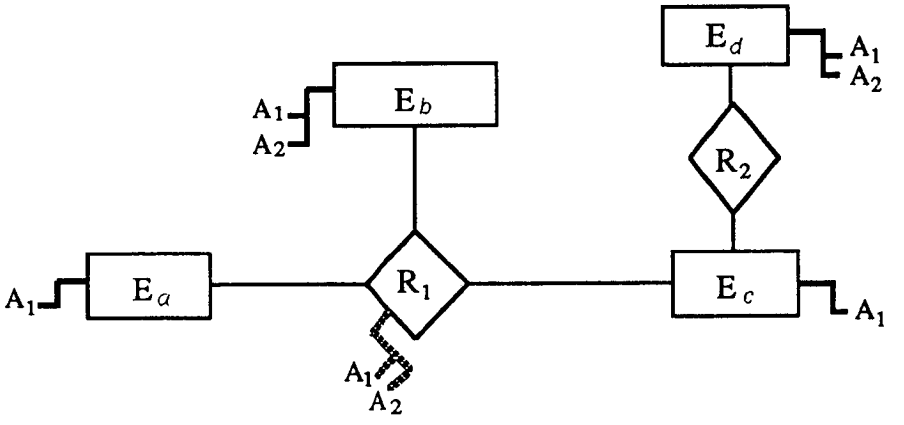
${}^{A}/_{Ei}$ sont des valeurs prises dans les domaines des attributs A des entités Ei, pour i = 1... mk, associées par R_k,

${}^{A}/_{Rk}$ sont des valeurs prises dans les domaines des attributs A de l'association R_k,

'[', ']' sont les symboles de début et de fin de phrase,

''' est le symbole qui entoure les attributs des associations.

exemple: le langage d'un modèle simple avec r = 2, m1 = 3 et m2 = 2,



$$L = \{\ [\ \text{nom\_de}(R_1)\ \ {}^{A1}/_{Ea}\ {}^{A1}/_{Eb}\ {}^{A2}/_{Eb}\ {}^{A1}/_{Ec}\ \text{''}\ {}^{A1}/_{R1}\ {}^{A2}/_{R1}\ \text{''}\ ]$$
$$[\ \text{nom\_de}(R_2)\ \ {}^{A1}/_{Ec}\ {}^{A1}/_{Ed}\ {}^{A2}/_{Ed}\ ]\ \}$$

figure 3

---

(1) Dans la théorie mathématique des langages:

- ε représente le mot vide
- K représente un mot
- K_1K_2 représente la concaténation de K_1 et de K_2, on a aussi $K\varepsilon = K$
- $K_1 + K_2$ représente le mot K_1 ou le mot K_2
- K^* represente $K^* = \bigcup_{i=0}^{\infty} K^i$, avec K^i succession de i concatenations de K

$$K^* = \varepsilon + K^1 + K^2 + K^3 + \ldots$$

- K^+ represente $K^+ = \bigcup_{i=1}^{\infty} K^i$

$$K^+ = K^1 + K^2 + K^3 + \ldots$$

Nous renvoyons pour plus de précisions à [DAVI87], [MANN74], [AHOH72], [AHOH73], [WIRT76], [SALO85], [AHOH74], [SALO73], dans notre ordre de préférence.

Une grammaire G context free du langage L est définie par un quadruplet $G = (\Sigma_T, \Sigma_N, P, s)$ où

- Σ_T est l'ensemble des symboles terminaux.
- Σ_N est l'ensemble de symboles non terminaux, on a $\Sigma_T \cap \Sigma_N = \phi$
- P est l'ensemble des règles de production de la forme $\alpha \rightarrow \beta$, où $\alpha \in \Sigma_N$ et $\beta \in (\Sigma_T + \Sigma_N)^*$ (2)
- s est la variable de départ de génération de phrases, $s \in \Sigma_N$.

Nous obtenons une grammaire $G = (\Sigma_T, \Sigma_N, P, s)$ où

$\Sigma_N = \{ \alpha, s \} \cup \{\alpha_k, \beta_k, \gamma_k / k = 1..r \}$

$\Sigma_T = \{ '[', ']', '"' \} \cup \{ nom\_de(R_k) / k = 1..r \} \cup \{'a'..'z'\} \cup \{'0'..'9'\}$

$P = \{$

$s \xrightarrow{1} '[' \alpha ']' s \mid \varepsilon$

$\alpha \xrightarrow{2} \alpha_k$, pour k = 1..r

$\alpha_k \xrightarrow{3} nom\_de(R_k)\ A/_{E1}\ A/_{E2} \ldots A/_{Emk}\ \gamma_k \beta_k$, pour k donné

$\beta_k \xrightarrow{4} A/_{E2} \ldots A/_{Emk}\ \gamma_k \beta_k \mid \varepsilon$, pour k donné

$\gamma_k \xrightarrow{5} '"' A/_{R_k} '"' \mid \varepsilon$, pour k donné

$\}$

La grammaire G génère les phrases du langage L. On dit encore que le langage de G est L.

exemple: on génère les phrases descriptives modèle simple précédent

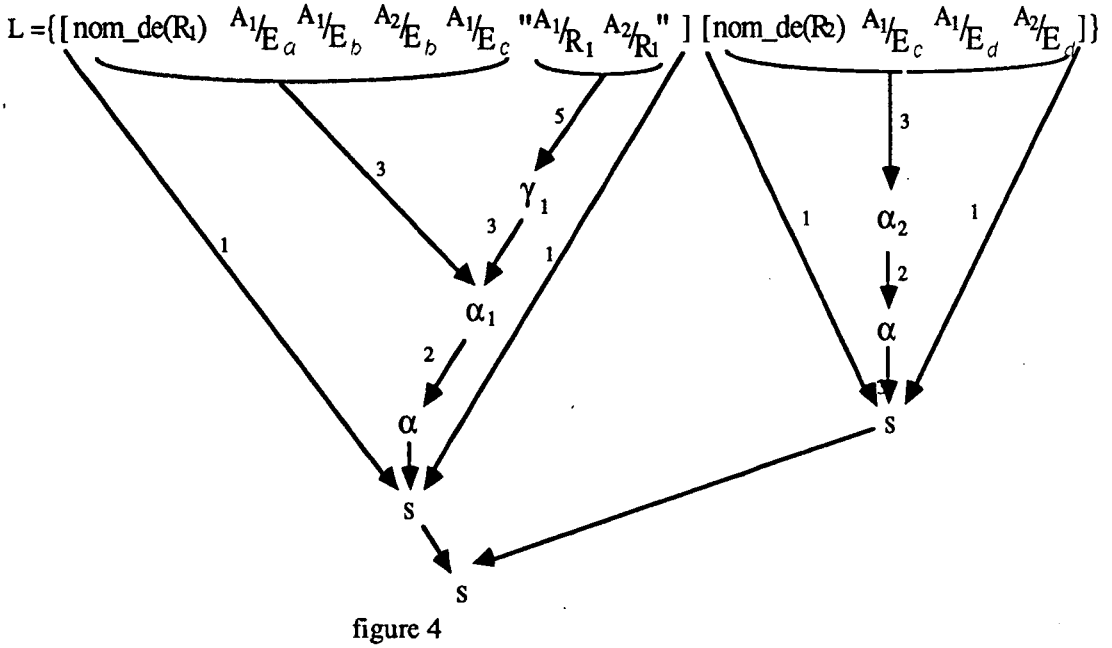


figure 4

Le lecteur trouve dans l'annexe II un exemple d'application de la grammaire à une association du schéma entités - associations défini sous I.2.

---

(2) est un mot constitué d'une suite de caractères pris dans l'ensemble Σ_T ou Σ_N.
cf aussi note (1).

II.2. Le langage du schéma des données du court terme

Nous ne donnons ici que la liste des phrases générées par la grammaire G pour notre schéma de données défini sous I.2.. L'annexe III fournit le programme de l'analyseur lexical-syntaxique de cette grammaire écrit avec les outils lex [3] et yacc [4].

Les occurrences d'associations d'entités $R_k \in \mathfrak{R}$ forment des phrases. Celles-ci sont délimitées par des crochets.

exemple: Une occurrence d'association *attache* entre entités REFERENCE et GAMME se note:

[*attache* **GAMME REFERENCE**]

Les noms d'associations d'entités R_k qui interviennent dans des phrases sont notés en caractères italiques.

exemple: nom_de(R_k) = *attache*

[*attache* **GAMME REFERENCE**]

Les noms d'attributs $A_l \in \mathcal{A}$ sur les associations R_k, i.e. $A_l : R_k \rightarrow D_l$, sont notés en petits caractères entre guillemets. Ils représentent les valeurs que l'attribut A_l prend lors d'une occurrence d'association R_k dans son domaine de base D_l.

exemple: l'attribut no_seq de l'association *composé* est noté "no_sequence", et no_sequence est une fonction, no_sequence : composé $\rightarrow \mathbb{N}$

[*composé* **GAMME OPERATION** "no_sequence" ... **OPERATION** "no_sequence"]

Les noms d'attributs $A_l \in \mathcal{A}$ sur les entités E_i, i.e. $A_l : E_i \rightarrow D_l$, sont représentés uniquement par le nom d'entité E_i en lettres majuscules. La grammaire substitue aux entités E_i l'ensemble des valeurs d'attributs qui lui sont associés.

exemple:

L'entité **OPERATION** représente **nom_op** qui prend des valeurs de chaînes de caractères.
TUNITE représente les attributs **nom_tunité** et **vitesse** qui prennent des valeurs de chaînes de caractère de réels, respectivement.

En reprenant l'ensemble des liens établis dans le schéma de données (figure 2 et 3) nous obtenons les phrases descriptives suivantes écrites dans la syntaxe du langage L.

• Phrases relatives à une gamme:

[*composé* **GAMME OPERATION** "no_sequence"... **OPERATION** "no_sequence"]

•

---

(3) lex est un générateur d'analyseur lexical

(4) yacc convertit une grammaire d'un langage context free dans un ensemble de tables utilisées par un algorithme LR(1) d'analyse syntaxique.

d'une opération par ses entrées et sorties

[*input* **OPERATION REFERENCE** "quantité" ... **REFERENCE** "quantité"]
[*output* **OPERATION REFERENCE** "quantité" ... **REFERENCE** "quantité"]

de l'obtention d'une référence de produit fini

[*attache* **GAMME REFERENCE**]

Les gammes sont linéaires et convergentes. Une référence obtenue par assemblage sera caractérisée par les gammes des références élémentaires qui la composent.

• Description des unités physiques et de leur regroupement sous forme de types:

les outils et types d'outils

[*appartient_tout* **OUTIL TOUTIL]**

les organes d'assistance et types d'organes d'assistance

[*appartient_torg* **ORGANE TORGANE]**

les machines et types de machines

[*appartient_tmach* **MACHINE TMACHINE]**

les unités et types d'unités

[*appartient_tunit* **UNITE TUNITE]**

• Description des stocks

[*situé_out* **STOCK OUTIL** "capacité"]
[*situé_org* **STOCK ORGANE** "capacité"]
[*situé_ref* **STOCK REFERENCE** "capacité"]

• Description des distances entre stocks et machines

[*distance* **STOCK** ***ou*** **MACHINE** **STOCK** ***ou*** **MACHINE** "longueur"]

• Les exécutions d'opérations sur des types de machines et les types d'outils et d'organes d'assistance, ainsi que les réglages nécessaires pour une opération donnée et les compétences qu'ils demandent sont décrits par:

[*exécuté* **OPERATION TMACHINE** "durée"]
[*nec_montage* **OPERATION TMACHINE TOUTIL**
COMPETENCE ... COMPETENCE "durée"]

[*nec_démontage* **OPERATION TMACHINE TOUTIL**
COMPETENCE ... COMPETENCE "durée"]

[*nec_montage* **OPERATION TMACHINE TORGANE**
COMPETENCE ... COMPETENCE "durée"]

[*nec_démontage* **OPERATION TMACHINE TORGANE**
COMPETENCE... COMPETENCE "durée"]

[*nec_réglage* **OPERATION TMACHINE**
COMPETENCE... COMPETENCE "durée"]

• Les individus, leurs compétences et leurs présences sont décrits par:

[*possède* **INDIVIDU COMPETENCE... COMPETENCE**]
[*présent* **INDIVIDU PERIODE**]

• Les transports de références sont décrits par:

[*transporte_ref* **UNITE REFERENCE** "qté_max"]
[*transporte_tout* **UNITE TOUTIL** "qté_max"]
[*transporte_torg* **UNITE TORGANE** "qté_max"]

On suppose que les unités de transports peuvent se déplacer entre tout stock et machine du système.

II.3. Description d'une gamme

A titre d'exemple nous décrivons une gamme de nom GAM1 (cf figure 5) qui permet d'obtenir une référence REFω. L'annexe IV fournit un texte descriptif étendu que nous avons soumis à l'analyseur lexical et syntaxique intégré donné en annexe III.

GAM1

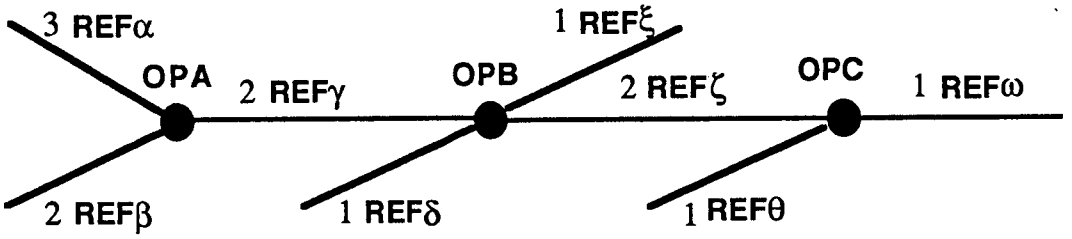


figure 5: représentation graphique de la gamme GAM1

La gamme GAM1 est rattachée à la référence REFω qui est le produit fini réalisé par cette gamme.

[*attache* **GAM1 REFω**]

La gamme se compose de trois opérations de nom OPA, OPB, OPC à exécuter dans cet ordre.

[*composé* **GAM1 OPA "1" OPB "2" OPC "3"**]

L'opération OPA nécessite 3 unités de la référence REFα, 2 unités de la référence REFβ et réalise 2 unités de la référence REFγ.

[*input* **OPA REFα "3" REFβ "2"**]
[*output* **OPA REFγ "2"**]

L'opération OPB nécessite 2 unités de la référence REFγ, 1 unité de REFδ et réalise 2 unité de REFζ et 1 unité de REFξ.

[*input* **OPB REFγ "2" REFδ "1"**]
[*output* **OPB REFζ "2" REFξ "1"**]

L'opération OPC nécessite 2 unités de la référence REFζ, 1 unité de REFθ et réalise 1 unité de REFω .

[*input* **OPC REFζ "2" REFθ "1"]**
[*output* **OPC REFω "1"]**

L'opération OPA peut s'exécuter sur un type de machine TM1 et nécessite une durée opératoire de 10 unités de temps. L'exécution d'OPA nécessite un type de d'organe d'assistance TORG1 monté sur TM1 en utilisant les compétences C1 et C3. Ce montage met 2 unités de temps. Le démontage de TORG1 met une unité de temps et nécessite la compétence C3 uniquement.

[*exécute* **OPA TM1 "00:10"]**
[*nec_montage* **OPA TM1 TORG1 C1 C3 "00:02"]**
[*nec_demontage* **OPA TM1 TORG1 C3 "00:01"]**

L'opération OPB s'exécute sur un type de machine TM2 et nécessite une durée opératoire de 15 unités de temps. L'exécution d'OPB utilise les types d'outils TOUT1 et TOUT2 dont le montage est réalisé en 6 unités de temps à l'aide des compétences C1, C2, C3 et C4. Le démontage de TOUT1 et TOUT2 met 2 unités de temps et nécessite la compétence C3 et C4 .

[*exécute* **OPB TM2 "00:15"]**
[*nec_montage* **OPB TM2 TOUT1 TOUT2 C1 C2 C3 C4 "00:06"]**
[*nec_demontage* **OPB TM2 TOUT1 TOUT2 C3 C4 "00:02"]**

L'opération OPC s'exécute sur un type de machine TM3 et met 5 unités de temps. L'exécution d'OPC nécessite un réglage d'une durée de 9 minutes par les compétences C1 et C4.

[*exécute* **OPC TM3 "00:05"]**
[*nec_réglage* **OPC TM3 C1 C4 "00:09"]**

Les machines physiques M1 et M2 sont du type TM1

[*appartient_tmach* **TM1 M1]**
[*appartient_tmach* **TM1 M2]**

La machine M3 est du type TM2

[*appartient_tmach* **TM2 M3]**

Les machines M4 et M5 sont du type TM3

[*appartient_tmach* **TM3 M4]**
[*appartient_tmach* **TM3 M5]**

Les outils OUT1 et OUT2 sont du type TOUT1.

[*appartient_tout* **TOUT1 OUT1]**
[*appartient_tout* **TOUT1 OUT2]**

Les outils OUT1 et OUT2 sont du type TOUT1.

[*appartient_tout* **TOUT2 OUT3]**

Les outils sont situés dans les stocks suivants:

OUT1 est situé dans le stock STOCK1 dont la capacité pour cet outil est de 5 unités

[*situé_out* **STOCK1 OUT1** "5"]

OUT2 est situé dans le stock STOCK2 dont la capacité pour cet outil est de 4 unités

[*situé_out* **STOCK2 OUT2** "4"]

OUT3 est situé dans le stock STOCK13 dont la capacité pour cet outil est de 6 unités

[*situé_out* **OCK13 OUT3** "6"]

Les organes d'assistance ORG1 et ORG2 sont du type TORG1. TORG1 est stocké dans STOCK3 et STOCK4 dont les capacités sont de 2 et de 3 unités pour ORG1 et ORG2, respectivement.

[*appartient_torg* **TORG1 ORG1**]
[*appartient_torg* **TORG1 ORG2**]
[*situé _org* **STOCK3 ORG1** "2"]
[*situé_org* **STOCK4 ORG2** "3"]

Les références sont situés dans les stocks suivants:

[*situé_ref* **STOCK5 REFα** "20"]
[*situé_ref* **STOCK6 REFβ** "10"]
[*situé_ref* **STOCK7 REFγ** "4"]
[*situé_ref* **STOCK8 REFδ** "8"]
[*situé_ref* **STOCK8 REFξ** "3"]
[*situé_ref* **STOCK10 REFζ** "5"]
[*situé_ref* **STOCK11 REFθ** "2"]
[*situé_ref* **STOCK12 REFω** "10"]

L'individu IND1 est présent entre 8:00 et 13:00 et possède les compétences C1 et C2

[*possède* **IND1 C1 C2**]
[*présent* **IND1** "08:00-13:00"]

L'individu IND2 est présent entre 8:00 et 13:00 et possède les compétences C2, C3 et C4.

[*possède* **IND2 C2 C3 C4**]
[*présent* **IND1** "08:00-13:00"]

L'individu IND3 est présent entre 13:00 et 21:00 et possède les compétences C1 et C5.

[*possède* **IND3 C1 C5**]
[*présent* **IND3** "13:00-21:00"]

II.4. Evaluation et developpements futurs

Nous venons de répertorier les entités intéressant la gestion du court terme et de représenter leurs liens à l'aide de la méthode simple du modèle entités - associations.

La définition formelle du modèle (dans I.1.) nous a permis de construire une grammaire context free (dans II.1.) d'un langage dont les phrases expriment de façon non redondante tous les liens que l'on établit dans le schéma de données. Cette grammaire générale a ensuite été reprise pour le schéma des données du système d'information du court terme (dans II.2.) et nous avons donné la liste des phrases qu'elle génère. Un exemple de description d'une gamme nous a permis de mettre en oeuvre le langage dans un contexte précis (dans II.3. et annexe IV).

Si la méthode utilisée a été simple et que le langage, par construction, répondait à la qualité de non redondance nous pouvons cependant retenir quelques insuffisances des résultats obtenus qui justifieront les efforts poursuivis dans les paragraphes suivants.

En tant que langage de description notre langage construit sur le modèle entités - associations se présente comme un ensemble de phrases à syntaxe immuable dont chacune ne décrit qu'une seule association. Ainsi on ne peut combiner des phrases en une seule décrivant une succession de liens existants.

exemple

1) Réunir des phrases:
dérire en une seule phrase les entrées et les sorties d'une opération

[OPERATION
input **REFERENCE** "quantité" **...** **REFERENCE** "quantité"
output **REFERENCE** "quantité" **...** **REFERENCE** "quantité"]

2) Substituer une phrase à un mot d'une phrase:
dans [*composé* **GAM1 OPA "1" OPB "2" OPC "3"**]
substituer à **GAM1** la phrase
[*attache* **GAM1 REFω**]
pour obtenir
[*composé* (*attache* **GAM1 REFω) OPA "1" OPB "2" OPC "3"**]

D'autre part par construction à partir du modèle entités - associations aucun opérateur de calcul n'intervient dans notre langage. Ainsi la description d'une gamme par une succession de phrases donne l'ensemble des temps opératoires de ses opérations, les temps de réglages et les temps de montages et de démontages, mais ne permet pas de calculer la somme de ces durées pour, par exemple, les comparer à celles d'autres gammes.

De même aucun opérateur ne permet de déduire d'une description d'une gamme à quelles références elle est rattachée, c'est - à - dire si la phrase [*attache* **GAM1 REFω**] est valide d'après la description de la gamme.

Un certain nombre de contraintes que l'on désire vérifier dans les descriptions faites à l'aide du langage peuvent être programmées par la structure des règles de production ou par une gestion des données de la phrase.

Par exemple, si l'on se donne la contrainte qui détermine que l'ensemble des références en entrée d'une opération ne peut être vide et leurs quantités sont non nulles.

$S \rightarrow$ '[' α ']'

$\alpha \rightarrow \alpha_k$

$\alpha_k \rightarrow$ *'input'* nom_op nom_ref γ_k β_k

$\beta_k \rightarrow$ nom_ref γ_k β_k | ε

γ_k	→	'"' quantité '"'
nom_ref, nom_op	→	CHAINE
quantité	→	ENTIERPOS
ENTIERPOS	→	'1' ENTIER | ... | '9' ENTIER
ENTIER	→	'0' ENTIER | ... | '9' ENTIER | ε
CHAINE	→	'a' SUITECHAINE | ... | 'z' SUITECHAINE
SUITECHAINE	→	'a' SUITECHAINE | ... | 'z' SUITECHAINE | '0' SUITECHAINE | ... | '9' SUITECHAINE | ε

La vérification de cette contrainte est syntaxique: la phrase qui décrit les entrées d'une opération doit, par définition des règles de production, donner au moins un nom de référence et une quantité nécessairement non nulle.

Une autre contrainte peut exiger qu'une gamme soit définie par une suite numérotée d'opérations dans laquelle chaque $i^{ème}$ opération pour i de 2 à n possède en entrée au moins une référence sortie de la $(i-1)^{ème}$ opération. Cette contrainte peut être verifiée par la gestion des données fournies par des phrases descriptives des entrées - sorties des opérations et de la gamme.

Ainsi toute contrainte retenue pour le système d'information doit être intégrée à l'analyseur lexical et syntaxique par voie de programmation.

Finalement, afin de répondre à l'exigence de cohérence des descriptions faites à l'aide du langage, il est indispensable de spécifier de façon cohérente les contraintes sur les entités, les associations et leurs attributs, et de définir les opérateurs de calcul dont nous venons de donner des exemples. Les modèles entités - associations permettent de rajouter au schéma de données un certain nombre de contraintes que nous pouvons partiellement inclure dans des règles de production d'une grammaire, mais ils ne proposent pas de méthode de calcul de la consistance c'est - à - dire de la non-contradiction de cet ensemble de contraintes et a forteriori de la consistance des descriptions faites sur le schéma de données.

Dans la suite de nos travaux nous nous orientons à la fois vers une méthode plus rigoureuse de spécification des types d'informations du système d'information et de définition opérationnelle des opérateurs de calcul dans cet univers ainsi que vers les méthodes de preuve automatique de cohérence d'une spécification.

Annexe I

Liste des entités, attributs, domaines et leur signification

Entités	Attribut	Domaine	Signification
GAMME	nom_gam	chaîne de caractères	nom ou code d'une gamme
OPERATION	nom_op	chaîne de caractères	nom ou code d'une opération
REFERENCE	nom_ref	chaîne de caractères	nom ou code d'une référence
UNITE	nom_unité	chaîne de caractères	nom ou code d'une unité physique de transport
TUNITE	nom_tunité	chaîne de caractères	nom ou code d'un type d'unité de transport défini par un intervalle de capacité de transport et de vistesse de déplacement
	vitesse	réel	vitesse de déplacement en km/h
STOCKS	nom_stock	chaîne de caractères	nom ou code d'un stockeur de références, d'outils, ou d'organes d'assistance
INDIVIDU	nom_ind	chaîne de caractères	nom ou matricule d'une personne participant au processus de production
COMPETENCE	nom_comp	chaîne de caractères	nom d'une compétence nécessaire pour les montages, démontages d'outils, d'organes d'assistance ou des opérations de réglage
OUTIL	nom_out	chaîne de caractères	nom ou code d'un outil physique
TOUTIL	nom_out	chaîne de caractères	nom ou code d'un type d'outil regroupant un ou plusieurs outils physiques en raison de caractéristiques techniques communes permattant une utilisation alternative des outils du même groupe.
ORGANE	nom_org	chaîne de caractères	nom ou code d'un organe d'assistance physique
TORGANE	nom_torg	chaîne de caractères	nom ou code d'un organe d'assistance regroupant un ou plusieurs organes d'assistance physiques en raison de caractéristiques techniques communes permattant une utilisation alternative des organes d'assistance du même groupe .
PERIODE	per	couple d'heures	durée de présence

Liste des associations, attributs, domaines et leur signification

Association	Attribut	Domaine	Signification
composé	no_seq	entier	associe des opérations à une gamme. Ces opérations sont numérotées dans l'ordre de leur exécution.
attaché	/	/	associe une référence de produit fini à une gamme qui permet de l'obtenir
input	quantité	réel	associe une référence à une opération. Cette référence appartient à l'ensemble des références utilisées en entrée de l'opération. On définit une quantité à utiliser pour chaque occurrence d'association.
output	quantité	réel	associe une référence à une opération. Cette référence appartient à l'ensemble des références en sortie de l'opération. On définit une quantité à utiliser pour chaque occurrence d'association.
exécuté	durée	temps	associe une opération à un type de machine sur lequel cette opération peut être effectuée. L'association précise la durée opératoire.
nec_montage	durée	temps	associe une opération à un type de machine sur lequel elle peut être effectuée ainsi que le montage d'un type d'outil ou d'organe d'assistance et les compétences nécessaires pour le montage. La durée du montage est donnée pour chaque occurrence d'association.
nec_démontage	durée	temps	associe une opération à un type de machine sur lequel elle peut être effectuée ainsi que le démontage d'un type d'outil ou d'organe d'assistance et les compétences nécessaires pour le démontage. La durée du démontage est donnée pour chaque occurrence d'association.
nec_réglage	durée	temps	associe une opération à un type de machine sur lequel elle peut être effectuée ainsi que les compétences nécessaires pour le réglage du type de machine. La durée du réglage est donnée.
appartient_tmach	/	/	associe des machines physiques à un type de machine en raison de caractéristiques opératoires communes qui permettent l'utilisation alternative des machines de ce type pour toute opération exécutable sur le type de machine.
appartient_tout	/	/	associe des outils physiques à un type d'outil en raison de caractéristiques techniques communes qui permettent l'utilisation alternative des outils de ce type.

appartient_tout	/	/	associe des organes d'assistance physiques à un type d'organes d'assistance en raison de caractéristiques techniques communes qui permettent l'utilisation alternative des organes d'assistance de ce type.
appartient_tunit	/	/	associe des unités de transport physiques à un type d'unité de transport en raison de caractéristiques techniques (capacité, vitesse de déplacement) communes qui permettent l'utilisation alternative des organes d'assistance de ce type.
transporte_ref *transporte_tout* *transporte_torg*	qté_max	réel	associe une référence, un type d'outil ou un type d'organe d'assistance à un type d'unité de transport. On précise à chaque occurence d'association la quantité maximale de référence, d'outil ou d'organe d'assistance que l'on peut transporter.
situé_ref *situé_out* *situé_org*	capacité	entier	associe une référence, un type d'outil ou un type d'organe d'assistance à un stock. On précise à chaque occurence d'association la capacité de stockage du stock en unités de référence, d'outil ou d'organe d'assistance qu'il peut contenir.
distance	longueur	réel	associe un stock à une machine, un stock à une autre stock ou une machine à une autre machine et indique la distance en mètres qui les sépare.
présence	/	/	associe un individu à une periode journalière de présence

Annexe II Exemple d'une phrase générée par la gramaire G

Dans le schéma entités - associations (I.2.) nous avons l'association:

Les règles de la grammaire G qui definit le langage de cette association:

$$s \longrightarrow \alpha$$

$$\alpha \longrightarrow nom\_gam\ \beta$$

$$\beta \longrightarrow nom\_op\ ''\text{'}no\_sequence\text{'}''\ \gamma$$

$$\gamma \longrightarrow \beta\ \gamma \mid \varepsilon$$

$$nom\_gam, nom\_op \longrightarrow \text{CHAINE}$$

$$no\_sequence \longrightarrow \text{ENTIER}$$

$$\text{CHAINE} \longrightarrow \text{'a' FINCHAINE} \mid \ldots \mid \text{'z' FINCHAINE}$$

$$\text{ENTIER} \longrightarrow \text{'1' FINENTIER} \mid \ldots \mid \text{'9' FINENTIER}$$

$$\text{FINCHAINE} \longrightarrow \text{'a' FINCHAINE} \mid \ldots \mid \text{'z' FINCHAINE} \mid \varepsilon$$

$$\text{FINENTIER} \longrightarrow \text{'0' FINENTIER} \mid \ldots \mid \text{'9' FINENTIER} \mid \varepsilon$$

On peut générer la phrase:

[*composé* **GAM1 OPA** "1" **OPB** "2"**OPC** "3"]

Annexe III

Nous avons écrit et implémenté un analyseur lexical et syntaxique intégré avec les outils lex et yacc pour le langage présenté sous II.2. (grammaire générale en II.1.). Nous donnons le programme de cet analyseur ci-après.

```
/*************************************************************************
*                                                                        *
*              L A N G A G E   de P R O D U C T I O N                    *
*                                                                        *
*                                 2/09/88                                *
*                                                                        *
*                                                                        *
*                     SOURCE DE L'ANALYSEUR LEXICAL                      *
*                     +++++++++++++++++++++++++++++                      *
*                                LEX                                     *
*                                +++                                     *
*                                                                        *
*       Les expressions régulières des token retournés à l'analyseur     *
*       syntaxique sont précédes d'un caractère "_".                     *
*       Les lignes du texte sont comptées et la valeur actuelle          *
*       du compteur est retournée à l'analyseur syntaxique pour la       *
*       localisation des erreurs.                                        *
*                                                                        *
*************************************************************************
*************************************************************************/

%{
        char tampon[20];
%}

%%
compose                 return(_COMPOSE);
input                   return(_INPUT);
output                  return(_OUTPUT);
attache                 return(_ATTACHE);
execute                 return(_EXECUTE);
nec_montage             return(_NEC_MONTAGE);
nec_demontage           return(_NEC_DEMONTAGE);
nec_reglage             return(_NEC_REGLAGE);
appartient_tmach        return(_APPARTIENT_TMACH);
appartient_tout         return(_APPARTIENT_TOUT);
appartient_torg         return(_APPARTIENT_TORG);
appartient_tunit        return(_APPARTIENT_TUNIT);
situe_out               return(_SITUE_OUT);
situe_org               return(_SITUE_ORG);
situe_ref               return(_SITUE_REF);
transporte_tout         return(_TRANSPORTE_TOUT);
transporte_torg         return(_TRANSPORTE_TORG);
transporte_ref          return(_TRANSPORTE_REF);
possede                 return(_POSSEDE);
present                 return(_PRESENT);
distance                return(_DISTANCE);
[0-9]+\.[0-9]+          return(_REEL);
\[                      return(_DEBUTPHRASE);
\]                      return(_FINPHRASE);
\"                      return(_GUILLEMETS);
\:                      return(_DOUBLEPOINTS);
```

```
\-                      return(_TIRET);
\n                      ligne++;
[0-9]+                  return(_ENTIER);
[a-zA-Z][a-zA-Z0-9]*    {
                                strncpy(tampon,yytext,19);
                                tampon[19] = '\0';
                                return(_CHAINE);
                        }
[ \t]+                  ;
\400                    return(_ENDFICH);

/*************************************************************************/
```

```
/*****************************************************************************
*****************************************************************************
*                                                                           *
*                        L A N G A G E       L A P O N                      *
*                                                                           *
*                                  2/09/88                                  *
*                                                                           *
*                                                                           *
*                        SOURCE DE L'ANALYSEUR SYNTAXIQUE                   *
*                        ++++++++++++++++++++++++++++++++                   *
*                                                                           *
*                                                                           *
*                                                                           *
*                                                                           *
*                                                                           *
*****************************************************************************
*****************************************************************************/


%{
#include <stdio.h>
int ligne = 1;

%}

%start  PHRASE
%token  _COMPOSE
%token  _INPUT
%token  _OUTPUT
%token  _ATTACHE
%token  _EXECUTE
%token  _NEC_MONTAGE
%token  _NEC_DEMONTAGE
%token  _NEC_REGLAGE
%token  _APPARTIENT_TMACH
%token  _APPARTIENT_TOUT
%token  _APPARTIENT_TORG
%token  _APPARTIENT_TUNIT
%token  _SITUE_OUT
%token  _SITUE_ORG
%token  _SITUE_REF
%token  _TRANSPORTE_TOUT
%token  _TRANSPORTE_TORG
%token  _TRANSPORTE_REF
%token  _POSSEDE
%token  _PRESENT
%token  _DISTANCE
%token  _TIRET
%token  _ENDFICH
%token  _CHAINE
%token  _ENTIER
%token  _REEL
%token  _GUILLEMETS
%token  _DEBUTPHRASE
%token  _FINPHRASE
%token  _DOUBLEPOINTS
%%

/*****************************************************************************/

PHRASE  :PHRASE _DEBUTPHRASE MOTS _FINPHRASE
        |
        ;

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

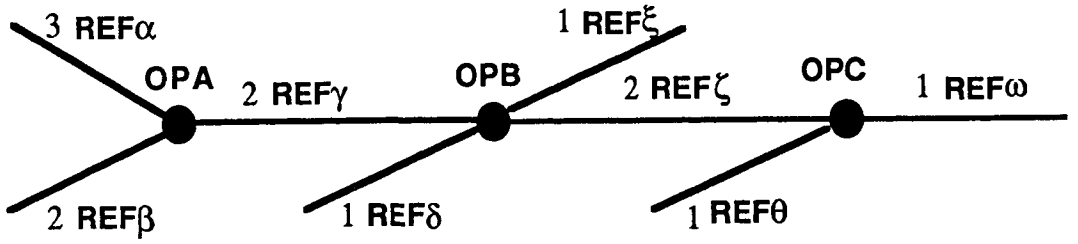
```
MOTS     :_COMPOSE _CHAINE OPE_SEQ

         |_INPUT    _CHAINE ENS_REF

         |_OUTPUT   _CHAINE ENS_REF

         |_ATTACHE  _CHAINE _CHAINE

         |_EXECUTE  _CHAINE _CHAINE _GUILLEMETS _ENTIER _DOUBLEPOINTS
                  _ENTIER _GUILLEMETS

         |_NEC_MONTAGE    _CHAINE _CHAINE ENS_MONTE ENS_COMP
                  _GUILLEMETS _ENTIER _DOUBLEPOINTS _ENTIER _GUILLEMETS

         |_NEC_DEMONTAGE _CHAINE _CHAINE ENS_MONTE ENS_COMP
                  _GUILLEMETS _ENTIER _DOUBLEPOINTS _ENTIER _GUILLEMETS

         |_NEC_REGLAGE    _CHAINE _CHAINE           ENS_COMP
                  _GUILLEMETS _ENTIER _DOUBLEPOINTS _ENTIER _GUILLEMETS

         |_APPARTIENT_TMACH      _CHAINE _CHAINE

         |_APPARTIENT_TOUT       _CHAINE _CHAINE

         |_APPARTIENT_TORG       _CHAINE _CHAINE

         |_APPARTIENT_TUNIT      _CHAINE _CHAINE

         |_SITUE_OUT     _CHAINE _CHAINE _GUILLEMETS _REEL _GUILLEMETS

         |_SITUE_ORG     _CHAINE _CHAINE _GUILLEMETS _REEL _GUILLEMETS

         |_SITUE_REF     _CHAINE _CHAINE _GUILLEMETS _REEL _GUILLEMETS

         |_TRANSPORTE_TOUT _CHAINE _CHAINE _GUILLEMETS _REEL _GUILLEMETS

         |_TRANSPORTE_TORG _CHAINE _CHAINE _GUILLEMETS _REEL _GUILLEMETS

         |_TRANSPORTE_REF  _CHAINE _CHAINE _GUILLEMETS _REEL _GUILLEMETS

         |_POSSEDE       _CHAINE ENS_COMP

         |_PRESENT       _CHAINE PERIODES

         |_DISTANCE      _CHAINE   _CHAINE _GUILLEMETS _REEL _GUILLEMETS
         ;

/*********************************************************************/

ENS_REF          :SUITE_REF _CHAINE _GUILLEMETS _REEL _GUILLEMETS
                 ;
SUITE_REF        :SUITE_REF _CHAINE _GUILLEMETS _REEL _GUILLEMETS
                 |
                 ;

/*********************************************************************/

ENS_COMP         :ENS_COMP _CHAINE
                 |
                 ;

/*********************************************************************/

PERIODES         :SUITE_PER _ENTIER _DOUBLEPOINTS _ENTIER _TIRET _ENTIER
                 _DOUBLEPOINTS _ENTIER
                 .
```

```
SUITE_PER       :SUITE_PER _GUILLEMETS _ENTIER _DOUBLEPOINTS _ENTIER
                 _TIRET  _ENTIER _DOUBLEPOINTS _ENTIER _GUILLEMETS
                |
                ;

/*********************************************************************/

ENS_MONTE       :SUITE_MONTE _CHAINE
                ;
SUITE_MONTE     :SUITE_MONTE _CHAINE
                |
                ;

/*********************************************************************/

OPE_SEQ         :SUITE_OP _CHAINE _GUILLEMETS _ENTIER _GUILLEMETS
                ;
SUITE_OP        :SUITE_OP _CHAINE _GUILLEMETS _ENTIER _GUILLEMETS
                |
                ;

/*********************************************************************/


%%

#include "lex.yy.c"

main()
{
        yyparse();
}

yyerror(s)
char *s;
{
        printf("\n erreur syntaxique %s \n LIGNE %d \n",s,ligne);
}
```

Annexe IV

Le texte descriptif de la gestion d'une usine produisant un produit fini REFω. Le texte a été vérifié syntaxiquement à l'aide de l'analyseur lexical et syntaxique présenté en annexe III.

GAM1




```
[ attache gam1 refomega ]

[ compose gam1 opa "1" opb "2" opc "3" ]

[ input  opa refalpha "3.0" refbeta "2.0" ]
[ output opa refgamma "2.0"]

[ input  opb refgamma "2.0" refdelta "1.0" ]
[ output opb refdseta "2.0" refxi "1.0"]

[ input  opc refdseta "2.0" reftheta "1.0" ]
[ output opc refomega "2.0" ]

[ execute opa tm1 "00:10"]
[ nec_montage opa tm1 torg1 c1 c3 "00:10"]
[ nec_demontage opa tm1 torg1 c3 "00:01"]

[ execute opb tm2 "00:15" ]
[ nec_montage opb tm2 tout1 tout2 c1 c2 c3 c4 "00:06"]
[ nec_demontage opb tm2 tout1 tout2 c3 c4 "00:02"]


[ execute opc tm3 "00:05" ]
[ nec_reglage opc tm3 c1 c4 "00:09"]

[ appartient_tmach tm1 m1]
[ appartient_tmach tm1 m2]
[ appartient_tmach tm2 m3]
[ appartient_tmach tm3 m4]
[ appartient_tmach tm3 m5]

[ appartient_tout tout1 out1]
[ appartient_tout tout1 out2]
[ appartient_tout tout2 out3]
[ appartient_tout tout2 out4]

[ appartient_torg torg1 org1]
[ appartient_torg torg1 org2]


[ situe_out     stock1  out1 "5.0" ]
[ situe_out     stock2  out2 "4.0" ]
[ situe_out     stock13 out3 "6.0" ]
```

```
[ situe_org     stock3  org1 "2.0" ]
[ situe_org     stock4  org2 "3.0" ]

[ situe_ref     stock5  refalpha        "20.0"  ]
[ situe_ref     stock6  refbeta         "10.0"  ]
[ situe_ref     stock7  refgamma        "4.0"   ]
[ situe_ref     stock8  refdelta        "8.0"   ]
[ situe_ref     stock8  refxi           "3.0"   ]
[ situe_ref     stock10 refdseta        "5.0"   ]
[ situe_ref     stock11 reftheta        "2.0"   ]
[ situe_ref     stock12 refomega        "10.0"  ]

[ possede ind1 c1 c2    ]
[ possede ind2 c2 c3 c4]
[ possede ind3 c1 c5    ]

[ present ind1  08:00-13:00 ]
[ present ind2  08:00-13:00 ]
[ present ind3  13:00-21:00 ]

[ transporte_tout tunit1 tout1 "5.0" ]
[ transporte_tout tunit1 tout2 "2.0" ]
[ transporte_torg tunit1 torg1 "4.0" ]
[ transporte_torg tunit13 torg1 "4.0" ]
[ transporte_ref  tunit2 refalpha       "2.0"  ]
[ transporte_ref  tunit2 refbeta        "1.0"  ]
[ transporte_ref  tunit3 refdseta       "11.0" ]
[ transporte_ref  tunit5 refgamma       "3.0"  ]
[ transporte_ref  tunit5 refxi          "7.0"  ]
[ transporte_ref  tunit2 refxi          "3.0"  ]
[ transporte_ref  tunit4 refdseta       "7.0"  ]
[ transporte_ref  tunit4 reftheta       "2.58" ]
[ transporte_ref  tunit4 refomega       "9.6"  ]


[ distance m1 m2 "7.2" ]
[ distance stock1 m1 "5.2" ]
[ distance stock1 stock2 "5.2" ]
```

REFERENCES BIBLIOGRAPHIQUES

[AHOH74] Aho, A., Hopcroft, J., Ullman, J., "The Design and Analysis of Computer Algorithms", Reading, Mass.: Addison Wesley, 1974.

[AHOH72] Aho, A., Ullman, J., "The Theory of Parsing, Translation and Compiling", vol. 1, Series in Automatic Computation, Prentice Hall, 1974.

[AHOH73] Aho, A., Ullman, J., "The Theory of Parsing, Translation and Compiling", vol. 2, Series in Automatic Computation, Prentice Hall, 1974.

[CHEN76] Chen, P., "The entity-relationship model-Towards a unified view of data", ACM Transactions on Database Systems 1, mars 1976, pp. 9-36.

[DAVI87] Davis, M.D., Weyuker, E.L., "Computability, Complexity and Languages, Fundamantals of theoretical Computer Science", Computer Science and applied Mathematics, Academic Press Inc, Oralando, Florida, 1987, p. 425.

[DILE88] Di Leva, A., Giolito, P., Vernadat, F., "Production System Specification: The M* Approach", Expert System in Manufacturing Design, (A. Kusiak,ed), SME, Dearborn, july 1988.

[ELAS85] Elamasri, R., Hevener, E., Weeldreyer, J., "The Category Concept: An Extension to the Entity - Relationship Model", Data Knowledge Engeneering, 1, 1, 1985, pp. 75-116.

[GALL84] Gallaire, H., "Techniques de compilation", Cepadues éditions, Toulouse, 1984, pp. 75-116.

[HAWR84] Hawryszkiewycz, I., "Database Analysis and design", SRA, Chicago,1984.

[MANN74] Manna, Z., "Mathematical Theory of Computation", McGraw Hill Computer Science Series, McGraw Hill, New York,1974.

[SALO85] Salomaa, A., "Computation and Automata", Cambridge University Press, Encyclopedia of Mathematics and its Applications vol. 25, 1985, p.282.

[SALO73] Salomaa, A., "Formal Languages", ACM Monographs Series, Academic Press,New York, 1973.

[SALO85] Salomaa, A., "Computation and Automata", Cambridge University Press, Encyclopedia of Mathematics and its Applications vol. 25, 1985, p.282.

[THEO86] Theorey, T.J., Yang, D., Fry, J.P., "A Logical Design Methodology for RelationnalDatabases Using the Extended Entity-Relationship Model", Computer Surveys, Vol.18,No 2, June 1986,pp.197-222.

[WIRT76] Wirth, N.,"Algorithms + Data Structures = Programs", Series in Automatic Computation, Prentice Hall, Englewood Cliffs,1976,p.366.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique