SIMULATION

D'INTERPRETEUR

LOGO

D'après les travaux du Laboratoire Didacticiel
du Centre Mondial et avec la collaboration des
équipes d'ACT Informatique

Une co-édition ACT Informatique/HATIER

Ce simulateur LOGO a été réalisé par le Laboratoire Didacticiel du Centre Mondial Informatique et Ressource Humaine et implanté sur THOMSON et Nanoréseau par les équipes de ACT Informatique.

Note sur la partie logicielle:
Elle a été développée initialement sur MS/DOS par Eric Bruillard et Nicole Rocland.
La mise au point est due à Eric Millet et Thierry Hénocque.

Note sur la présente documentation:
La rédaction du présent ouvrage a été assurée par Thierry Marchaisse et Thierry Hénocque sur la base d'une première version écrite par Eric Bruillard et Nicole Rocland.

Avertissement

ISBN 2-7201-1404-X Co-édition ACT Informatique/HATIER

Pour lancer le logiciel

Sous nanoréseau

Une fois chargé le MS-DOS vous chargez le nanoréseau en tapant NR31<ENTREE>. Le logiciel se chargera automatiquement. Un menu s'affiche sur votre MO5. Tapez le caractère "C" qui correspond à l'option LOGO1.0.

Sur votre TO7 70 ou votre MO5

Une fois LOGO chargé il vous faut taper RAMENE "DEPART <ENTREE>.

Qu'est-ce qu'un interpréteur ?

Tout le monde a une idée, plus ou moins précise, de ce qu'est un ordinateur. C'est une machine qui dispose d'une "mémoire" et qui peut exécuter certaines actions élémentaires d'une manière très rapide, comme faire la somme de deux nombres, envoyer un caractère sur un écran ou un périphérique, etc... La programmation consiste à grouper ces actions élémentaires pour en faire des actions plus complexes. Si on en reste au niveau de la machine, les programmes que l'on peut créer restent très proches des actions de base, et leur conception reste l'apanage d'un petit nombre de specialistes. C'est pourquoi on adjoint à la machine un traducteur, ou un interpréteur dont la tâche consiste à rendre exécutable par la machine des programmes écrits dans une langue plus proche de l'utilisateur (et aussi plus proche du problème que celui-ci veut traiter).

Ce logiciel a pour but d'expliciter le processus par lequel un interpréteur permet l'exécution de phrases (ou si l'on préfère de formules) complexes. Pour cela,il faut décomposer cette machine qu'est un interpréteur en plusieurs parties .Pour l'instant,nous distinguerons deux parties dans notre interpréteur (plus tard nous affinerons notre analyse) :

(1) un exécuteur , qui connait un certain nombre d'actions de base. Il ne peut théoriquement en exécuter qu'une seule à la fois. La liste de ces actions correspond dans ce logiciel à la liste des primitives reconnues par la simulation.

Shématiquement on peut imaginer, que chaque action ou opération correspond à un micro-programme chacun de ces micro-programmes étant totalement indépendant des autres [1].

On remarquera que L'exécuteur est ainsi vu comme une boite noire . Mais il est hors de question d'expliciter ici le fonctionnement de cette sous-machine , notre documentation n'y suffirait pas.

> (2) une machine à traduire groupant plusieurs éléments, et dont la fonction consiste à décomposer phrases et formules afin de les transformer en action élémentaires accéssibles à l'exécuteur.

Evidement la "compétence" de l'exécuteur détermine et donc limite les mots initiaux qui sont à la disposition de l'utilisateur.Un groupement quelconque de ces mots initiaux n'a pas forcément un sens. L'interprèteur correspond tout à fait à une grammaire: il a des règles de syntaxe très précises à respecter et qui sont liées à la grammaire que l'on a choisie.Une syntaxe rigide est une condition (formelle) sine qua non pour que parmi toutes les phrases qu'on peut former à partir des mots initiaux certaine aient un sens.

Ce logiciel est centré essentiellement sur les diverses manipulations à effectuer pour que l'exécuteur puisse travailler.

[1] Bien sur, ceci ne correspond pas du tout à ce qui se passe effectivement en machine,mais une telle shématisation n'a rien d'alarmant sur le plan théorique.

Sur quoi peut-on agir ?

Quand on charge le logiciel, on est dans un mode, fixé par défaut, qui est décrit complètement dans les manuels d'utilisation et de référence.

Ce mode initial n'est pas figé.On peut intervenir directement sur le modèle proposé et changer le mode d'interprètation et donc la grammaire du langage .

Les utilisateurs ayant déjà une bonne connaissance de Logo disposent ainsi d'un logiciel ouvert, sur lequel ils peuvent changer:

- les mots initiaux (rajouter, enlever des primitives)

- le fonctionnement local de chaque partie de la machine

- le fonctionnement global en créant une procédure d'interprètation automatique

Tout ce qui concerne la représentation interne des objets, i.e. la manière dont l'ordinateur stocke et récupère les divers types d'objets qu'il a à gérer, est laissé de côté. En particulier, les problemes d'adressage ne sont pas abordés.Cela signifie que l' on conservera, tout au long de cette documentation,un point de vue naif. On supposera notamment une accession directe à tous les objets.

Ce logiciel doit être vu avant tout comme une approche concrète (via LOGO) de la notion d'interprète. On notera qu' il constitue par la même une bonne introduction à la programmation en langage LOGO, puisqu' il en décrit de l'interieur le fonctionnement.

Nous vous souhaitons bonne chance dans votre exploration du monde des interpréteurs.

SOMMAIRE

GUIDE D'UTILISATION

Introduction

ATTENTION: Ceci est une simulation, toute ressemblance avec des versions du langage Logo existantes est purement fortuite...

Le programme que nous allons étudier simule l'interprètation d'une ligne d'ordres Logo. Il semble plus facile tout d'abord de se restreindre aux ordres qu'on peut construire avec des primitives arithmétiques. Les primitives graphiques posent en effet un problème pratique pour notre simulation

(on verra que l'on ne peut visualiser simultanément notre modèle d'interprèteur et l'exécution d'une primitive graphique).

Cette restriction n'est d'ailleurs que provisoire: une fois bien comprise l'interprètation d'une ligne arithmétique telle que:

EC SOMME LISCAR PRODUIT HASARD 12 8

l'extension de notre interprèteur aux primitives graphiques, (en général beaucoup plus simples) ne pose aucun problèmes.

Petite note sur la lecture de ce guide:

Avant d'essayer de se servir du programme de simulation, il est conseillé de lire le chapitre "Démarrage de la simulation". Le chapitre suivant: "Exemples d'utilisation", devra être lu parralèlement à son application avec la simulation si l'on veut voir illustrer les explications qui y sont données.

Le mot primitive est utilisé dans plusieurs sens différents. Il peut désigner

- soit les primitives du Logo ,
- soit les mots considérés comme des primitives par la simulation,
- soit les procédures qui constituent le logiciel et qui se manipulent comme de véritables "primitives " (on emploie parfois le terme de macro-primitives).

Dans chaque cas, le contexte vous permettra de comprendre dans quelle acception le terme "primitive" est utilisé.

Démarrage de la simulation

Mise en route du système

Mettez la cassette "SIMULATION D'INTERPRETEUR LOGO" dans votre lecteur de cassette et tapez :

RAMENE "DEPART et envoyez avec la touche "ENTREE"

Vous allez voir apparaitre sur votre écran le message suivant:

Patientez quelques instants il faut charger le programme . . .

Le chargement du programme est terminé lorsque le message

Tapez DEMO pour une demo, DEBUT sinon.

apparait. Ne vous impatientez pas trop, ce chargement est relativement long.

Pour avoir une première idée du fonctionnement du simulateur, le mieux est de lancer tout d'abord une démonstration en tapant DEMO.

Après un petit instant vous lirez sur l'ecran :

Tapez la liste d'instructions à exécuter, sinon tapez ""EXO"":

(Tapez ? pour avoir la liste des primitives connues de la simulation

Le programme vous invite à choisir une ligne afin de l'interpréter. Vous pouvez soit choisir vous-même cette ligne, soit laisser cette initiative au programme[4]

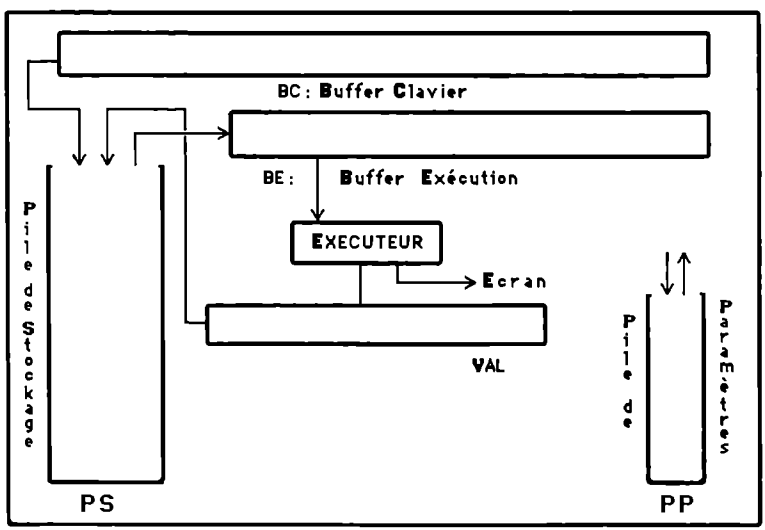
[4] Voir à ce sujet la partie concernant les différents exercices proposés dans le chapitre sur les exemples d'utilisation.

Si vous tapez, par exemple:

EC SOMME 3 4

(et envoyez avec la touche "ENTREE")
Vous devez alors voir sur l'écran :

Représentation de l'interpréteur sur L'ecran :



En regardant ce qui se passe sur l'écran, vous allez voir comment procède notre simulateur pour exécuter la phrase d'entrée. Des messages en bas de l'écran vous précisent les différentes actions effectuées.

Tout ce qui est "interne" est représenté sur l'écran et un bip sonore accompagné d'un message vous préviennent au moment des exécutions

(ces dernières peuvent conduire aussi à un affichage en bas de l'écran , c'est le cas avec la primitive EC par exemple).

Une fois le programme lancé toute réutilisation de l'interprèteur commence par le message suivant :

tapez la liste d'instruction à exécuter

Pour revoir en détail et à votre rythme la séquence que vous venez de faire effectuée, vous pouvez utiliser la commande REPRENDS. Cette séquence va de nouveau être exécutée, mais en "pas à pas". Si vous vous placez dans ce mode n'oubliez pas qu'il vous faut à chaque fois taper sur une touche quelconque pour déclencher l'action (élémentaire) suivante. Le nom de cette dernière s'affiche en bas de l'écran , puis elle est exécutée.

Si vous le souhaitez, vous pouvez ensuite relancer une autre démonstration, sinon vous entrez dans le vif du sujet,auquel cas vous tapez :

DEBUT.

Comme tout à l'heure, le programme vous demande une liste d'instructions. Entrez par exemple:
EC PROD 5

Cette ligne est alors stockée dans le buffer clavier (BC) et vous avez de nouveau la main. A vous d'explorer les différentes commandes à votre disposition pour interprèter correctement cette suite de caractères .

Il n'y a en fait que 5 actions :

TFCS, TFSE, EXEC, RETOUR et FINI

(la pile de paramètres PP étant gérée automatiquement par le système).

Vous trouverez ci-après quelques indiquations indispensables pour utiliser le programme DEBUT et comprendre les sigles: TFCS etc. Pour plus de détails reportez vous au manuel de référence.

La commande ANNULE vous permet de supprimer la dernière action que vous avez lancée il est donc inutile de tout recommencer si vous pensez avoir suivi une fausse piste.

Une autre facon de s'en sortir est d'utiliser REPRENDS qui va vous permettre de revoir la suite des actions que vous avez commandées.

En tapant sur la touche "S" (pour Stop), vous reprenez la main quand vous le désirez.

A tout moment, la commande INTERPRETE vous permet de demander au système de finir l'interprétation de la ligne en cours á votre place:

du moins celui-ci va-t-il tenter de le faire à partir de l'état dans lequel vous aurez mis votre simulation , c'est-à-dire au moment ou la procèdure INTERPRETE est appelée.

Voici brièvement récapitulées quelques-unes des principales actions élémentaires de la simulation :

DEBUT	initialise et lance la simulation
TFCS	TransFère le premier élément du buffer Clavier dans la pile de Stockage
TFSE	TransFère le premier élément de la pile de Stockage dans le buffer d'Exécution
EXEC	Donne le contenu du buffer d'exécution à l'exécuteur
RETOUR	transfère la valeur contenue dans VAL dans la pile de stockage

FINI	déclare que l'interprètation de la ligne introduite au début est terminée
INTERPRETE	demande au système de finir le traitement de la ligne introduite au début
DEMO	lance le mode de démonstration
ANNULE	annule la dernière action effectuée
REPREND	reprend une séquence d'instructions à partir du début en pas à pas
DESSIN	reconstitue le graphisme de la simulation
?	récapitule brièvement les primitives de la simulation

Exemples d'utilisation

" Interpréter EC SOMME 4 5 "

Imaginons que vous souhaitiez "décortiquer" l'éxécution de la liste d'instructions:

EC SOMME 4 5.

Pour initialiser cette simulation, tapez tout d'abord DEBUT: BC, PS, PP, BE, VAL [1] sont alors vidés de tout contenu (s'il y en avait un), et la simulation vous sollicite par le message:

Tapez la liste d'instructions à éxécuter,
sinon tapez "EXO":
(Tapez "? " pour avoir la liste des
primitives connues de la simulation)

Tapez alors (dans notre exemple):

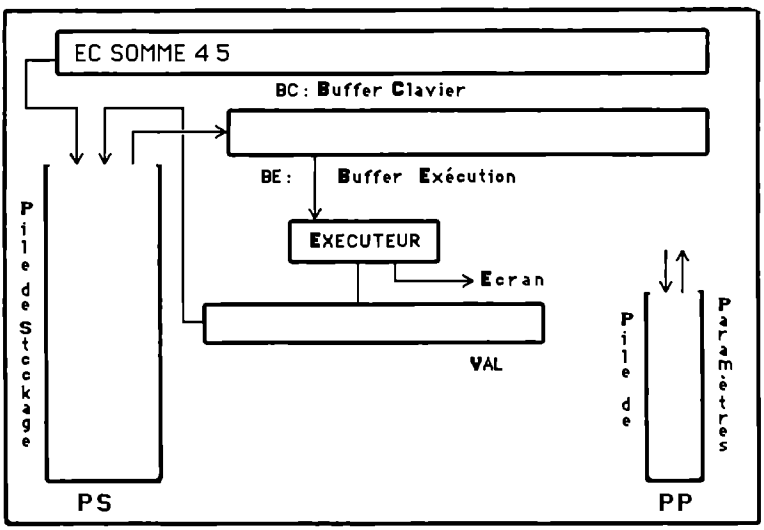
EC SOMME 4 5.

La simulation affiche dans BC ce que vous venez de taper et vous rend la main (vous voyez apparaitre le "?" du Logo en début de ligne).

Remarque: dans ce chapitre, nous avons joint au texte les écrans que l'on rencontre, après chaque action, dans le traitement de la phrase précedente, ceci pour permettre au lecteur d'en suivre le déroulement sans être nécéssairement en face d'une machine. Pour bien comprendre, il est cependant préférable de taper réellement les différents ordres et de controler leur action sur l'écran. Si vous faites une faute d'orthographe en tapant un nom, cela n'a aucune importance. Retapez le convenablement, la simulation n'a pas été altérée.

[1] Si vous n'avez pas d'ordinateur jetez un coup d'oeil sur les premiers paragraphes du manuel de référence pour vous familiariser avec la signification des diverses abréviations: BC,PS etc.

l'écran graphique après avoir tapé la première ligne



Remarque:
Les mots trops longs pour les cadres seront tronqués à la longueur maximum de ce qu'ils peuvent contenir.Un point indique alors que le mot (ou la phrase) est incomplet.

Vous disposez d'un ensemble de primitives [2] pour interpréter cette ligne:

- des actions élémentaires:
  TFCS, TFSE, EXECU, RETOUR, FINI;
- des renseignements sur les primitives:
  PRIMI?, NPAR, PRIM;
- des retours en arrière:
  ANNULE, REPREND;
- deux types de visualisation:
  CACHE, VISIBLE;

A tout moment, vous pouvez taper "?" pour avoir des renseignements sur ces primitives.

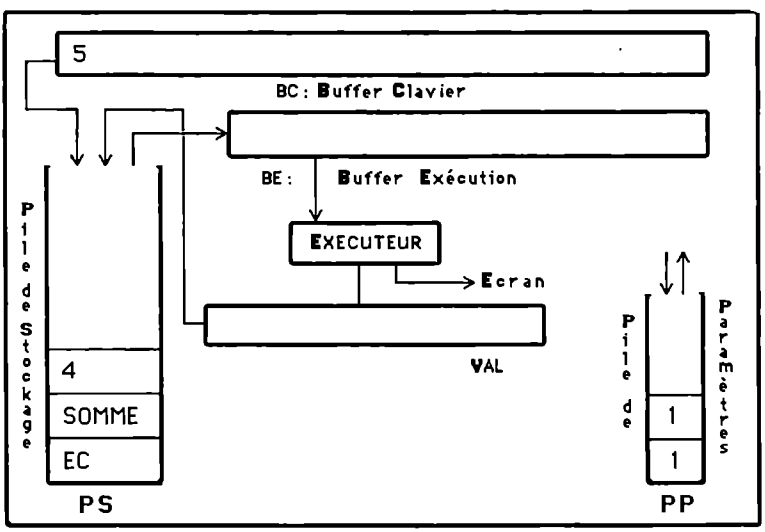
Allons-y:

Vous tapez:	**Commentaires:**
TFCS	Le premier élément de BC (ici l'ordre"EC") est tranféré dans PS. Durant ce transfert, la simulation examine l'élément transféré. Il s'agit ici d'une primitive qui attend un argument, le nombre 1 est donc empilé dans PP. A la fin de cet ordre, la simulation est dans l'état suivant: BC: SOMME 4 5
TFCS	SOMME est transféré de BC à PS. C'est une primitive qui attend 2 arguments, nombre qui est empilé dans PP. PP contient alors 1 et 2, indiquant par la que la dernière primitive empilée dans PS attend encore 2 arguments, et que l'avant-dernière en attend encore 1. A la fin de cet ordre, la simulation est dans l'état suivant: BC: 4 5 PS: EC SOMME BE: VAL: PP: 1 2
TFCS	Le nombre 4 est transféré dans PS: c'est un nombre, donc a priori un argument d'une primitive précédemment transférée dans PS, laquelle attend donc un argument de moins. Ceci est matérialisé par le sommet de PP qui est décrémenté: (dans notre cas, il passe de 2 a 1) .

A la fin de cet ordre, la simulation est dans l'état suivant:

BC: 5
PS: EC SOMME 4
BE:
VAL:
PP: 1 1

Apres l'ordre TFCS :



TFCS 5 est tranféré de BC a PS. C'est une valeur numérique, donc un argument d'une primitive, donc le sommet de PP est décrémenté. A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: EC SOMME 4 5
BE:
VAL:
PP: 1 0

Le 0 en sommet de PP indique que la dernière primitive transférée dans PS a tous ses arguments et qu'elle peut donc être évaluée. Il reste alors à tranférer un à un les éléments de PS dans BE jusqu'au transfert de cette dernière primitive.

TFSE 5 est tranféré de PS à BE. A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: EC SOMME 4
BE: 5
VAL:
PP: 1 0

TFSE 4 est transféré de PS à BE. A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: EC SOMME
BE: 4 5
VAL:
PP: 1 0

TFSE La primitive SOMME est transférée de PS à BE. Quand on transfère une primitive de PS à BE cela conduit à l'exécution de cette primitive, qui doit avoir tous ses arguments, et donc à la fin du traitement de cette primitive.

Le sommet de PP (qui indique le nombre d'arguments encore attendus par la primitive, dans notre cas 0) est donc dépilé. A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: EC
BE: SOMME 4 5
VAL:
PP: 1

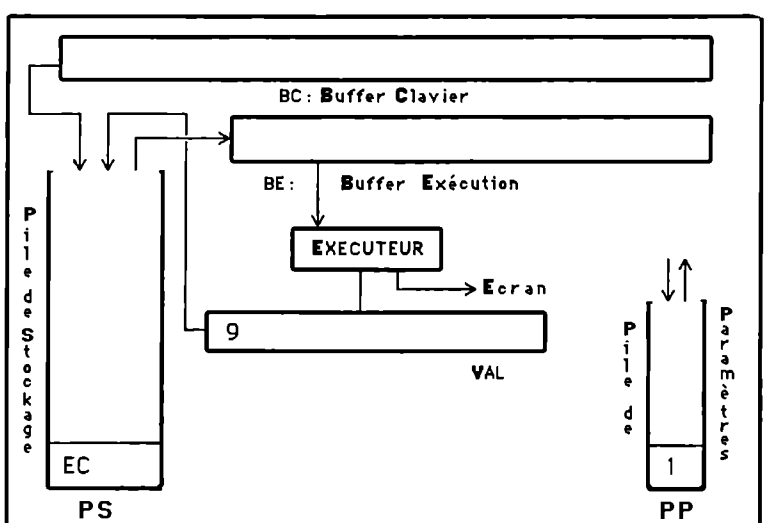
EXECU Cet ordre transmet à l'exécuteur Logo le contenu de BE. Après exécution, le résultat est transmis :
- soit à l'écran (c'est le cas des primitives graphiques ou de certaines primitives de texte comme par exemple EC)

- soit stocké dans une case mémoire nommée VAL dans cette simulation. Dans notre exemple,le résultat de l'exécution de SOMME 4 5, à savoir 9, est stocké dans VAL.

A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: EC
BE:
VAL: 9
PP: 1

Après l'ordre EXECU



RETOUR Cet ordre provoque le transfert du contenu de VAL dans PS.Le contenu de VAL qui est le résultat de l'exécution d'un ordre Logo, est

à priori considéré comme un argument de la "plus haute" primitive contenue dans PS (en fait celle qui est le plus proche du sommet de PS). Le sommet de PP est donc décrémenté. A la fin de cet ordre, la simulation est dans l'état suivant:

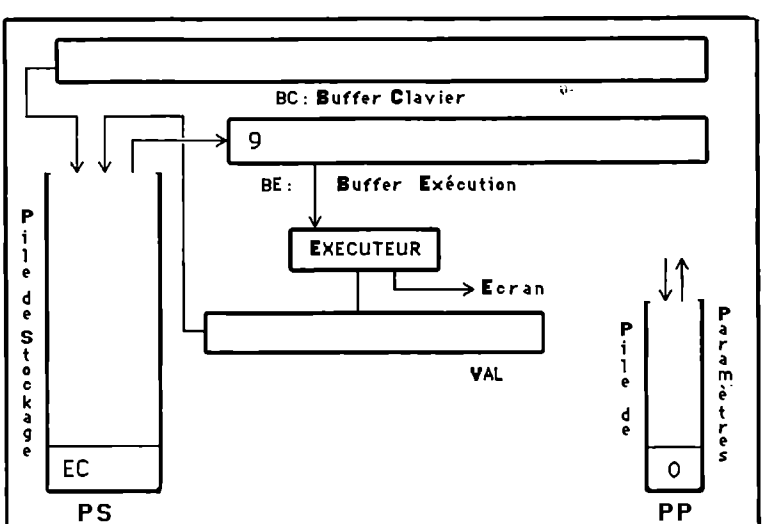
BC:
PS: EC 9
BE:
VAL:
PP: 0

TFSE

Le sommet de PP valant 0, la primitive "la plus haute" dans PS a tout ses arguments, et l'on transfert dans BE jusqu'à obtention de celle-ci. A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: EC
BE: 9
VAL:
PP: 0

Après l'ordre TFSE:



TFSE La primitive EC est transférée dans BE. Le sommet de PP est dépilé. A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS:
BE: EC 9
VAL:
PP:

EXECU On vient de tranférer une primitive dans BE, on peut donc envoyer le contenu de BE à l'exécuteur Logo. Le résultat de cette exécution provoque l'affichage du nombre 9 sur l'écran texte. A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS:
BE:
VAL:
PP:

FINI Cet ordre controle l'état des différents buffers et piles.En cas de problème il donne un diagnostic à l'utilisateur.Mais on remarquera que les messages envoyés par FINI sont en fait très succints: "O.K." indique que tout semble correct; "Il reste des choses à faire" indique que quelque chose "cloche".

Un simple coup d'oeil à l'écran suffit alors pour se rendre compte du problème.

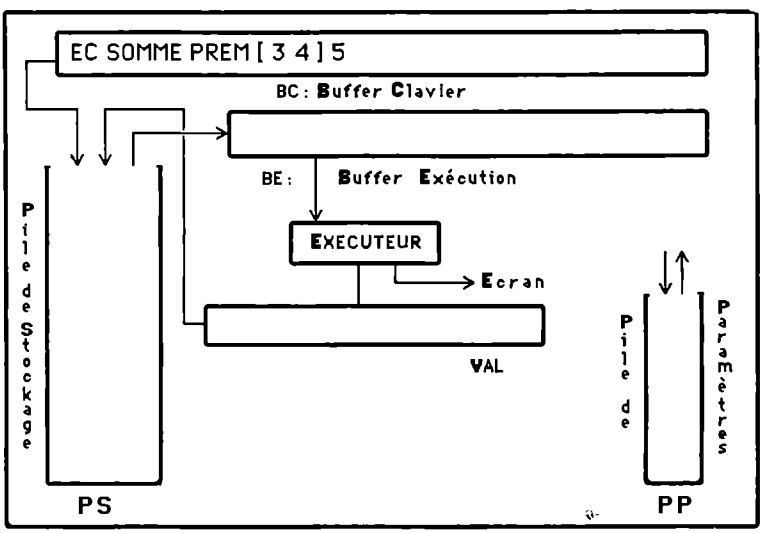
Dans le cas présent comme nous avons (sans rencontrer de problèmes) mené notre simulation à bien, le programme écrit "O.K"

"Interpreter EC SOMME PREM [3 4] 5"

Voici, plus brièvement commenté que dans l'exemple précédent, un autre exemple de simulation. Réinitialisez la simulation en tapant: DEBUT

Maintenant tapez EC SOMME PREM [3 4] 5.

L'écran après avoir tapé la première ligne:



Vous tapez: **Commentaires:**

REPETE 4 [TFCS]

Comme nous sommes toujours "sous Logo",nous pouvons bénéficier de toutes les primitives standards du Logo et nous dispenser ainsi de répétitions fastidieuses.
EC, puis SOMME, puis PREM, puis [3 4] sont transférés de BC à PS.
A la fin de cet ordre, la simulation est dans l'état suivant:

BC: 5
PS: EC SOMME PREM [3 4]
BE:
VAL:
PP: 1 2 0

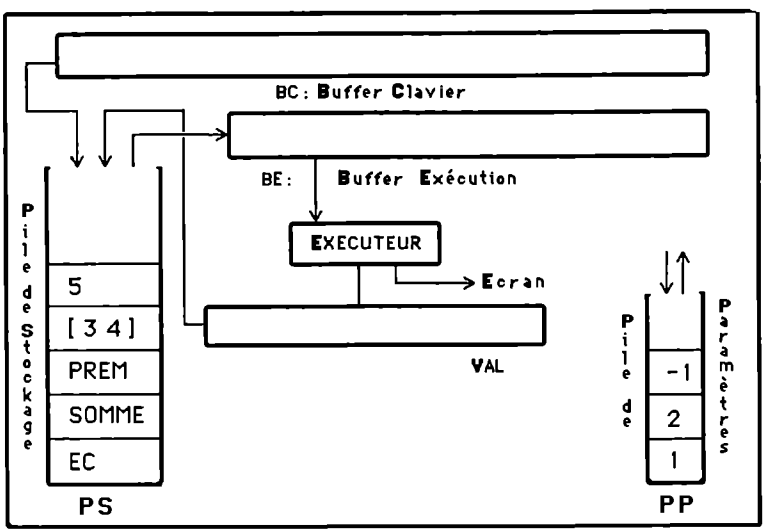
TFCS

Emporté par le feu de l'action, nous venons de taper une bêtise! A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: EC SOMME PREMIER [3 4] 5
BE:
VAL:
PP: 1 2 -1

D'une manière générale, voir un nombre négatif dans PP est signe d'une erreur par rapport au fonctionnement normal. Dans un cas de ce genre il nous faut donc annuler notre dernière action.

Après l'ordre TFCS:



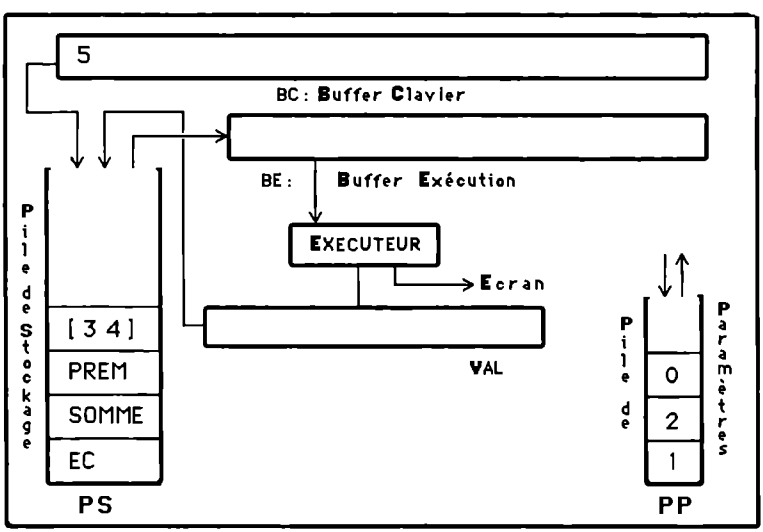
Un débordement de l'une des deux piles PS ou PP provoquera aussi une vision partielle, l'élément du bas de la pile étant alors remplacé par des points de suspension indiquant qu'il y à encore des choses "en-dessous".

ANNULE — On annule notre action précédente qui s'avère ne pas convenir du tout a la situation. Une autre manière de rattraper notre erreur aurait pu être de taper REPREND, de taper 4 fois sur la barre d'espace pour provoquer les 4 premiers TFCS, puis d'appuyer sur la touche S pour interrompre le REPREND. On se retrouve alors dans l'état souhaité.

Après l'ordre ANNULE le message "A vous " s'affiche sur l'écran et la simulation est dans l'état suivant:

BC: 5
PS: EC SOMME PREM [3 4]
BE:
VAL:
PP: 1 2 0

Après l'ordre ANNULE:



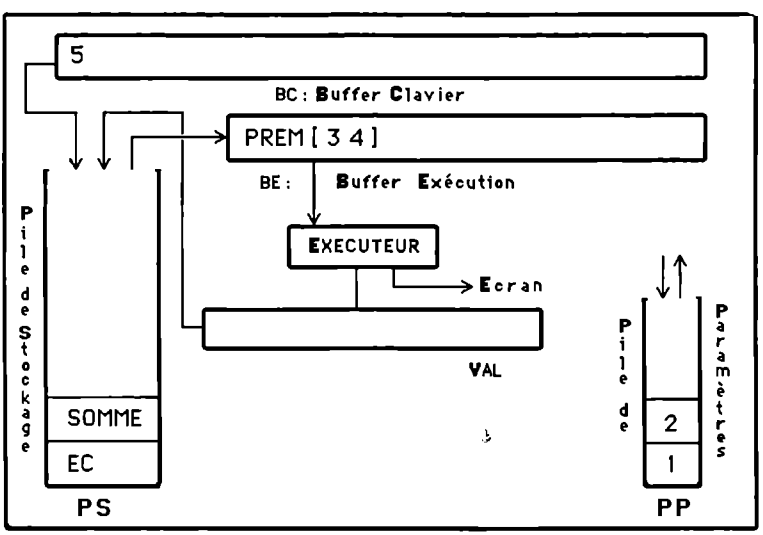
TFSE La liste [3 4] est transférée de PS à BE. A la fin de cet ordre, la simulation est dans l'état suivant:

BC: 5 PS: EC SOMME PREM
BE: [3 4]
VAL:
PP: 1 2 0

TFSE La primitive PREM est transférée de PS à BE. Le sommet de PP est dépilé. A la fin de cet ordre, la simulation est dans l'état suivant:

BC: 5
PS: EC SOMME
BE: PREM [3 4]
VAL:
PP: 1 2

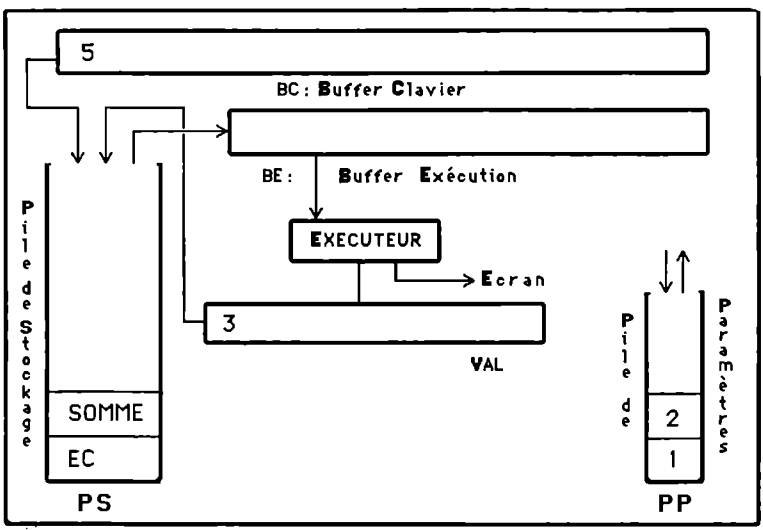
Après l'ordre TFSE:



EXECU Le contenu de BE est transmis à l'exécuteur Logo. A la fin de cet ordre, la simulation est dans l'état suivant:

BC: 5
PS: EC SOMME
BE:
VAL: 3
PP: 1 2

Après l'ordre EXECU:



RETOUR 3 est transféré de VAL à PS. A la fin de cet ordre, la simulation est dans l'état suivant:

BC: 5
PS: EC SOMME 3
BE:
VAL:
PP: 1 1

CACHE Pour s'amuser, mais aussi pour apprendre à raisonner dans le cadre de notre interpréteur on décide de continuer. A partir de maintenant,le traitement en aveugle!!!
Seuls restent visibles le contenu de PP et les messages explicatifs. Toutes les autres expressions sont remplacées par des petits pavés.

A la fin de cet ordre, la simulation est dans l'état suivant:

BC: X
PS: X X X
BE:
VAL:
PP: 1 1

TFCS Le sommet de PP étant à 1, il reste un argument à transmettre à une primitive. VAL étant vide, ce ne peut donc être qu'un argument en provenance de BC. A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: X X X X
BE:
VAL:
PP: 1 0

TFSE Le sommet de PP vaut 0, il faut donc commencer à transférer le contenu de PS dans BE, jusqu'à l'obtention d'une primitive. A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: X X X
BE:X
VAL:
PP: 1 0

EC PRIMI? PREM :BE

On teste si l'on vient de transférer une primitive.

La réponse est : FAUX !!

Il faut donc continuer à transférer.On notera qu'on aurait pu se passer de ce test en remarquant que lors du transfert d'une primitive de PS à BE, le sommet de PP est dépilé. En guettant la pile PP, on peut donc savoir à quel moment le premier élément de BE est une primitive.Ce type de test est cependant indispensable dès que PP n'est plus géré automatiquement.

TFSE A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: X X
BE: X X
VAL:
PP: 1 0

EC PRIMI? PREM :BE

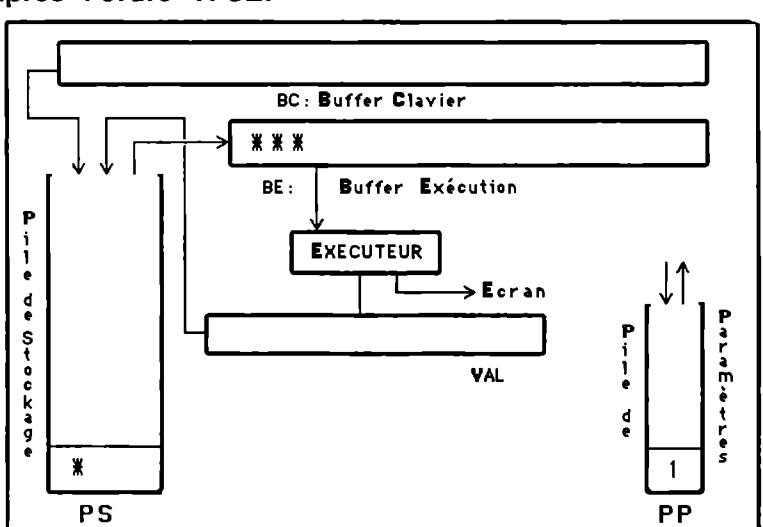
On teste encore une fois si l'on vient de transférer une primitive. La réponse est encore : FAUX !!

Il faut donc continuer à transférer.

TFSE Le sommet de PS est transféré dans BE.
A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: X
BE: X X X
VAL:
PP: 1

Après l'ordre TFSE:



EC PRIMI? PREM :BE

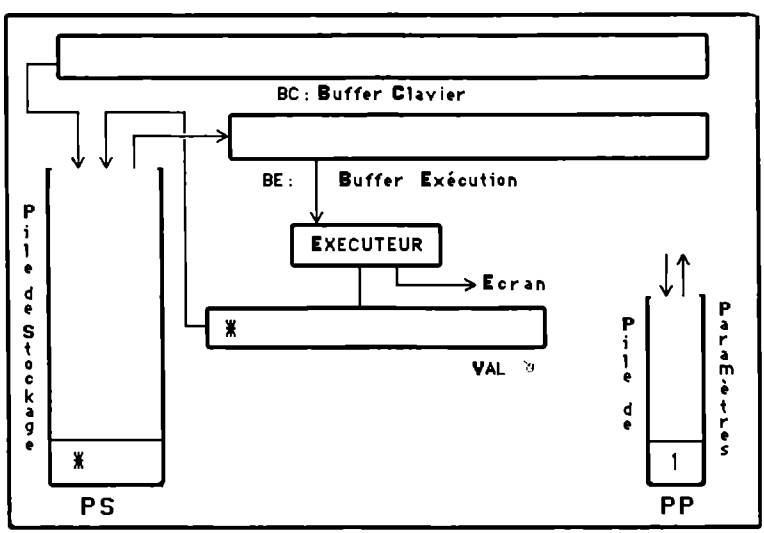
On teste si l'on vient de transférer une primitive.

La réponse est cette fois ci: VRAI !!! (Le sommet de PP vient d'être dépilé) On va donc pouvoir lancer l'exécution .

EXECU — Le contenu de BE est transmis à l'exécuteur LOGO. A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS: X
BE:
VAL: X
PP: 1

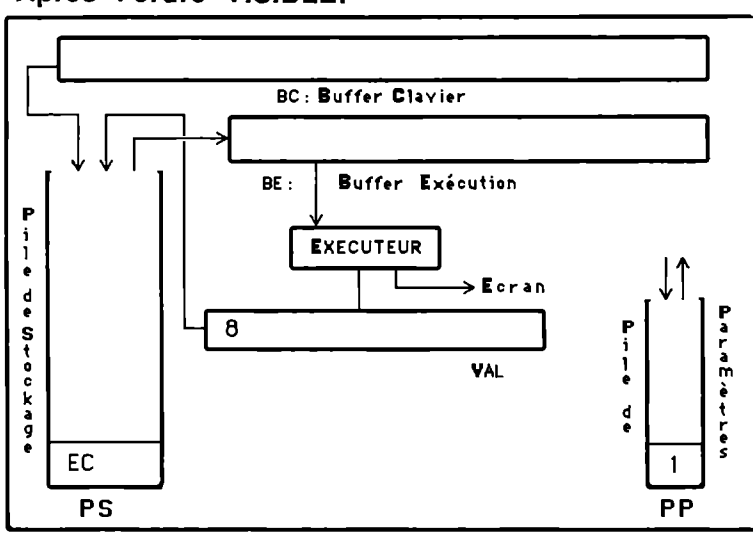
Après l'ordre EXECU:



VISIBLE On revient à une situation directement "lisible par l'oeil humain"... A la fin de cet ordre, la simulation est dans l'état suivant:

BC:
PS:EC
BE:
VAL: 8
PP: 1

Après l'ordre VISIBLE:



INTERPRETE On donne la main à la simulation pour qu'elle termine toute seule l'interprétation (on la voit alors se dérouler sur l'écran).

L'option EXO

Après le fichier "INTERPRE" ,il y a une série de fichiers "EXERCICE" qui sont chargés lorsque l'on tape "EXO" (comme phrase à interpréter).

Vous pouvez soit les prendre dans l'ordre soit positionner la cassette à un endroit quelconque avant de lancer EXO !

Remarque :

Il y a toute sorte d'exercices . Il y a même des exercices dont la macro-primitive "INTERPRETE" ne viendrait pas à elle seule à bout. A vous de trouver le moyen de les interpréter (i.e de chercher quelle grammaire permettrait d'aboutir à l'exécution correcte de la phrase).

Quelques lignes à exécuter

Voici différents exemples de lignes LOGO que vous pouvez essayer d'interpréter à l'aide de cette simulation.

1 EC SOMME 3 4 EC 2

2 EC SOMME PRODUIT 4 5 DIFF 10 2

3 EC SOMME HASARD 10 PRODUIT 3 5

4 EC HASARD SOMME PRODUIT 2 3 4

5 EC 3 EC SOMME 1 3 EC 4

Si on utilise REPRENDS, la fonction HASARD est réexécutée, et on peut donc obtenir un résultat différent.
Vous pouvez aussi introduire des lignes du même type.

Remarque
Si vous introduisez une longue ligne du type
EC SOMME SOMME 3 PRODUIT 2 SOMME 4 5 7
il est possible que vous oubliiez des données. Mais comme tout se passe au niveau supérieur de Logo, vous pouvez toujours rajouter des choses dans le buffer clavier.
Dans notre exemple, BC étant vide, PS contenant [EC SOMME 4] et BE étant vide, il est impossible de faire exécuter la ligne (il manque une donnée pour SOMME).
Vous pouvez alors écrire DONNE "BC [3], et continuer l'interprétation. (Ce changement ne sera pas pris en compte si vous utilisez ensuite ANNULE ou REPRENDS, à moins que vous ne tapiez aussi
DONNE ".BC MD "3 :.BC
Car .BC est la copie de ce que contenait BC au démarrage)

Répertoire de quelques pièges

Pour interpréter une ligne donnée, l'utilisateur a à sa disposition diverses actions et il peut les utiliser à sa guise. En fait, tout est permis, le système ne testant à aucun moment la piste suivie. On peut donc se permettre
quelques "élucubrations" comme par exemple:

(a) "Exécuter plusieurs primitives à la fois"

Prenons la ligne:
EC SOMME 1 3

Un premier détournement va consister à faire exécuter toutes les primitives par LOGO en stockant tout dans BE via PS.Pour obtenir cela on tape tout simplement

REPETE 4 [TFCS] REPETE 4 [TFSE].

On tente d'exécuter, mais le système refuse en envoyant le message:

"Exécution impossible. Il y a trop de primitives"

En effet, l'exécuteur n'accepte de faire son office que si on lui envoie une seule primitive avec tous ses arguments
correctement évalués. Ceci empêche de faire interpréter une ligne directement par LOGO et non par la simulation.

(b) "Tout transférer dans PS

On peut aussi tout empiler dans PS, puis dépiler au fur et à mesure dans BE pour exécuter les primitives une à une,mais une phrase comme "EC 2 EC 5" sera alors exécutée à l'envers. On aura successivement 5 puis 2 au lieu de 2 suivi de 5.

Pour la suite, on supposera que l'utilisateur essaye de se débrouiller "réellement" avec la simulation afin de découvrir par lui même quelques pièges classiques.

(c) "Employer TFSE trop tot"

Si on reprend la ligne: EC SOMME 1 3, on peut procéder comme suit: :

Ordre	contenu des variables			
	BC	PS	BE	PP
TFCS	[SOMME 1 3]	[EC]		[1]
TFCS	[1 3]	[EC SOMME]		[1 2]
TFCS	[3]	[EC SOMME 1]		[1 1]
TFSE	[3]	[EC SOMME]	[1]	[1 1]
TFCS	[ ]	[EC SOMME 3]	[1]	[1 0]
TFSE	[ ]	[EC SOMME]	[3 1]	[1 0]

A partir de là on peut continuer sans problème, aucune erreur ne sera detectèe et on aura le résultat correct (4) affiché sur l'écran .

Ce qui est "frauduleux" ici c'est d'avoir commencé à transférer de PS à BE sans que le sommet de PP soit égal à 0.L'exécution similaire de EC DIFF1 3 conduit à un résultat faux (on obtient 2 et pas -2). Cette façon de procéder n'est, bien sur, pas généralisable à d'autres phrases.Le résultat était correct sur l'exemple précédent uniquement par ce que la primitive SOMME était commutative, i.e. l'ordre de ses deux arguments n'intervenait pas sur le résultat final.

(d) "Faire un TFCS avant un retour"

C'est une erreur similaire à la précédente. Elle consiste à ne pas effectuer un RETOUR tout de suite après l'exécution d'une primitive qui retourne et à faire un transfert (TFCS ou TFSE) avant.

Sur une phrase comme EC SOMME PREM [2 3] 5 nous devrons à un moment exécuter SOMME 2 5. Si dans le cas d'opérations commutatives (comme SOMME) cela ne semble pas très génant, cela pose tout de suite un problème avec des primitives comme DIFF ou MOT...

(e) "La débrouillardise"

Un dernier détournement enfin, consiste à jongler avec les primitives de la simulation pour arriver coûte que coûte à interpréter une ligne en fait ininterprétable.

Considerons la phrase#:

EC EC 3 4

On peut faire successivement TFCS, TFCS, TFCS, TFSE, TFSE EXECU.

A ce moment, "3" est écrit sur l'écran , et on a la situation suivante:

BC: [4]
PS: [EC]
BE: [ ]
VAL: [ ]
PP: [1]

Théoriquement, le sommet de PP étant 1, il faut transférer dans PS soit le contenu de VAL, soit le premier élément de BC.

Effectuer un RETOUR n'est pas possible, VAL étant vide. On effectuera donc un TFCS qui empilera 4 dans PS et permettra l'exécution ultérieure.

Le sommet de PP étant alors décrémenté, il devient égal à 0. On peut donc terminer cette exécution (bien que la ligne ne soit pas à priori exécutable).

On peut trouver autant qu'on voudra de lignes LOGO inexécutables et qu'on peut néanmoins traiter avec cette simulation. En voici quelques exemples:

```
   SOMME 1 3 EC
(on transfère EC avant d'effectuer le
   RETOUR du 4)

   1 EC
(on transfère 1 dans BE puis on s'occupe de EC)

   LISCAR EC SOMME 1[2]
(on transfère EC SOMME 1 avant
d'effectuer le RETOUR du caractère lu).
```

Bricoler l'interpréteur : portée et limite du procédé

Une des caratéristiques de notre interpréteur est qu'aucun contrôle n'est effectué par la simulation sur la suite des ordres tapés par l'utilisateur.

En conséquence de très nombreuses phrases LOGO théoriquement ininterprétables y sont pratiquement tout à fait interprétables. De même on a vue qu'il est possible, lorsque l'interprétation d'une phrase LOGO correcte a été mal commencée, de continuer malgré tout son intreprétation en se debrouillant pour "rattraper le coup".

Considérées du point de vue d'un "vrai" interpréteur les diverses caractéristiques du notre seraient des défauts majeurs! Mais notre but n'est pas de fournir ici un logiciel techniquement impeccable.Notre but est de fournir un environnement de type "interpréteur" qui permette à l'utilisateur:

(a) de bien visualiser les problèmes informatiques que posent l'interprétation d'expressions dont la syntaxe n'est pas correcte.

(b) de comprendre la manière dont peut s'y prend un interpréteur (un vrai) pour traiter une ligne d'ordres.

Nous espèrons que notre esquisse d'interpréteur est assez simple et assez séduisante pour amener ceux qui la manipuleront à vouloir l'enrichir voire même à tenter de créer leur propre syntaxe.

Utilisation du mode CACHE

Quand vous passez dans ce mode, il est plus intéressant d'essayer d'interpréter des lignes choisies par le système (elles vous sont ainsi totalement inconnues).
Tapez DEBUT et au lieu d'introduire une ligne par vous-mëme tapez EXO .

Chaque mot constituant la ligne est remplacé par un petit pavé. Vous devez donc interpréter cette ligne sans savoir exactement quels sont les objets que vous manipulez.

La gestion de la pile de paramètres étant automatique, le transfert du buffer clavier vers la pile de stockage ne pose aucun problème.
Par contre, dès que vous devez transférer quelque chose dans le buffer d'exécution, il vous faut savoir à quel moment vous arrêter donc pouvoir tester ce qu'il y a dans BE...
EC PRIMI? PREM :BE
vous retournera VRAI ou FAUX suivant que le premier élément du buffer d'exécution est une primitive connue du système ou non. C'est alors à vous d'utiliser correctement cette information.

Une manière plus simple, et déjà signalée, consiste à observer la pile PP. Au cours du transfert de PS à BE, seules les primitives ont une action sur PP, cette action consistant à ôter le sommet de cette pile. Ainsi, dès que PP est dépilée, vous êtes sür d'avoir transféré une primitive dans le buffer d'exécution.

Si en cours de route vous voulez voir "en clair" ou vous en ètes, vous pouvez taper VISIBLE, qui vous permettra de visualiser l'état de la simulation et éventuellement de continuer ainsi.

Tout en restant en mode CACHE, vous pouvez revoir tout ce que vous avez fait ,si vous en avez besoin, par la commande REPRENDS.Vous pouvez aussi utiliser ANNULE ou INTERPRETE .

·Travailler en mode CACHE vous permet de vérifier la méthodologie que vous avez dégagée pour interpréter des lignes LOGO est correcte.

Une primitive difficile: LISCAR

Notre interpréteur traite la primitive LISCAR à part, son fonctionnement est en effet un peu spécial.
Essayez d'interpréter la ligne:

EC LISCAR

LISCAR n'attend aucun paramètre (NPAR "LISCAR retourne 0, c'est une primitive sans argument). En effet, ce n'est qu'au cours de son exécution que cette primitive va demander à l'utilisateur un argument en requérant de celui-ci qu'il tape sur une touche.
Une fois LISCAR chargé dans le buffer d'exécution (BE), tapez EXECU. En bas de l'écran on peut lire alors:

J'execute: LISCAR
J'attends un chiffre [1]

Tapez alors par exemple sur la touche "5", le 5 s'affiche sur l'écran texte et la simulation vous dit "Merci". Cette valeur est transferée dans VAL, et l'interprétation continue.

[1] Dans cette simulation, nous avons choisi pour simplifier la compréhension de la primitive LISCAR, de ne l'employer que pour la lecture de chiffres. Cela évite, en particulier en mode CACHE, des "erreurs" sur des lignes comme EC SOMME 3 LISCAR .Tous nos exemples, toutes les lignes à interpréter que nous proposons, supposent que LISCAR saisit un nombre.
Remarque: La simulation n'effectuant aucun controle sur le caractères que vous tapez au clavier, vous pouvez cependant taper n'importe quel caractère si vous le souhaitez.

Le programme conserve la valeur que vous avez donnée, pour pouvoir la réutiliser si un ordre REPRENDS) intervient.
Vous pouvez vous entrainer sur de nombreuses lignes utilisant la commande LISCAR. Par exemple:

EC SOMME SOMME LISCAR LISCAR LISCAR

EC PRODUIT SOMME 2 LISCAR DIFF 4 LISCAR

EC LISCAR EC HASARD LISCAR

De la même manière, la primitive LL n'attend aucun argument. à son exécution, elle provoque une lecture au clavier, et le résultat de son exécution sera stocké dans VAL.

Remarque: on dit d'ordinaire que LISCAR "va chercher" un caractère au clavier.Cela peut créer une confusion avec ce qu'on appelle Buffer Clavier dans la simulation. Il faut bien voir que ce dernier contient une suite d'ordres tapés au clavier et envoyés (en appuyant sur la touche "ENTREE") à l'interpréteur qui la stocke,l'analyse puis l'exécute. Une fois que l'on a appuyé sur la touche "ENTREE" il n'y a plus de rapport entre cette suite et le clavier.

Les variables

Cette simulation ne serait pas possible sans la création de nouvelles primitives.Il y a tout d'abord ASSOCIE qui est équivalente à DONNE .Il y a aussi OBJET qui correspond à CHOSE (CONTENU peut sembler meilleur, mais c'est un mot réservé sur certaines versions de LOGO).

La gestion des variables dans une simulation comme la notre pose un problème incontournable.Il est indispensable en effet de pouvoir différencier les noms crées par l'utilisateur de ceux dont le programme à besoin pour travailler: d'où les procédures ASSOCIE et OBJET qui permettent d'effectuer cette distinction.

L'utilisation de ces divers synonymes offre l'avantage de pouvoir travailler aussi bien en mode direct, sous LOGO, que par l'intermédiaire de notre mini-interpréteur. On peut ainsi taper, sans passer par DEBUT ou DEMO des ordres comme:

```
ASSOCIE "X 12
EC OBJET "X
```

Exemple:

Après avoir exécuté ASSOCIE "TOTO 50 (soit en mode direct, soit en utilisant la simulation), on peut tester :

```
ASSOCIE "TATA SOMME :TOTO 40   [1]
```

Puis en mode direct :

```
EC OBJET "TATA
```

Vous devez voire apparaitre sur l'écran :

```
90
```

[1] La même ligne en mode direct ne fonctionnera pas correctement, car le (:) est un caractère réservé de LOGO qui va donc chercher l'objet associé à TOTO, sans consulter la liste VAR. Cela marchera toujours en remplacant (:) par OBJET.

Remarque:

REPRENDS ne restitue pas la suite des affectations qu'on aura été amené à opérer.En effet, ceci nous obligerait à dupliquer la variable VAR avant chaque séquence. Ainsi, si au départ VAR contient [TOTO 50 TATA 30] et que vous exécutez la ligne:

ASSOCIE "TOTO SOMME :TOTO 40 à

l'issue de l'interprétation, on aura dans VAR:

[TOTO 90 TATA 30].

Une nouvelle exécution avec REPRENDS conduirait à lire dans VAR:

[TOTO 130 TATA 30].

Vous pouvez bien sur, à la main, restaurer les valeurs intiales, en tapant en mode direct: ASSOCIE "TOTO 50 avant d'utiliser REPRENDS ou ANNULE.

Les mots

Les atomes littéraux

On dispose des commandes classiques: MOT, PREM, DER, SP et SD.

L'interprétation de ces diverses commandes reste la même dans notre mini-interpréteur, on peut donc enrichir de ces ordres les lignes sur lesquelles on travaille. En voici quelques exemples :

```
ASSOCIE MOT "TO "TO SD "TITI

ASSOCIE PREM "TOTO SOMME PREM 34 LISCAR
```

Les mots et les nombres

Une petite parenthèse sur les nombres s'impose ici. Qu'est-ce que 5? C'est à la fois un chiffre (i.e. un symbole lexical) et un nombre (i.e. la valeur du symbole). Lorsque l'on parle de nombres, on fait sans arrêt des aller et retour entre ces deux points de vue. Ainsi, faire l'addition de 5 et de 3 revient à les considérer comme valeur. En revanche affirmer que "pour multiplier un nombre par 10 il suffit de lui ajouter un 0 à droite" revient à considerer ce nombre et le 0 comme des symboles lexicaux (il en est de même pour les classiques critères de divisibilité.On dira par exemple "un nombre est divisible par 5 s'il se termine par un 0 ou un 5".Mais c'est évidement du chiffre qu'on veut parler...).

LOGO n'échappe pas à cette dualité. Ainsi, le test EGAL? "12 12, dans lequel 12 est successivement considéré comme chaine de caractères puis comme valeur, retourne VRAI.

On aboutit ainsi à l'écriture d'expressions qui semblent d'une cohérence plutot douteuse:

```
EC SOMME MOT "12 4 6
```

et qui sont néanmoins parfaitement admises par LOGO.D'autre part dès qu'ils dépassent une certaine taille les "nombres" ne peuvent plus (pour LOGO) être considérés autrement que comme des chaines de caractères.

Ainsi, pour le LOGO MO5 une expression comme

```
EC SOMME MOT "123456789 12 12
```

retournera un message d'erreur,le nombre étant trop grand.

On peut aussi noter qu'un nombre ne peut pas ëtre employé comme nom de procédure.

Les listes

Dans notre simulation d'interpréteur, les listes sont vues d'une manière très simpliste, i.e. comme un tout:
elles sont transférées globalement. Pour les traiter, on ne dispose au départ que des primitives fonctionnant aussi sur les mots: PREMIER, DERNIER, SP, SD.

Si vous le souhaitez, vous pouvez ajouter aux primitives déjà connues de la simulation d'autres primitives [1] permettant de traiter les listes (PHRASE, LISTE, MP, MD, ...).

Les primitives SI et EXECnécessitent un traitement spécial. Elles ne sont donc pas incluses dans la version de base de la simulation.

[1] Voir à ce sujet le chapitre AJOUTER DES PRIMITIVES.du guide d'utilisation

Les booléens

VRAI et FAUX sont considérés ici comme des mots ou, plus exactement, comme des "variables système": il faut donc les faire précéder du caractère (") On dispose des connecteurs logiques OU, ET, NON et de EGAL? [1]
Voici quelques exemples:

EC EGAL? 10 SOMME 4 LISCAR

EC OU ET EGAL? 3 LISCAR "VRAI EGAL? 2 HASARD 5

EC NON OU NON EGAL? 2 HASARD 4 EGAL? 1 LISCAR

[1] Ici encore, l'utilisateur pourra ajouter les prédicats qu'il souhaite tester, comme par exemple : LISTE?, MEMBRE?, INF?, SUP?, MOT?, NOMBRE?, VIDE? etc...

Ajouter des primitives

Pour enrichir cette simulation, l'utilisateur peut ajouter d'autres commandes:

- soit des primitives du Logo (comme VIDE?, MP, etc..)

- soit des procédures qui seront considérées par la simulation comme des primitives [1]

La primitive de la simulation AJPRIM (pour AJoute une PRIMitive) permet d'ajouter de nouvelles primitives pour la durée d'une séance de travaille.

Syntaxe : AJPRIM Nom

L'ordinateur vous demande alors si la primitive retourne une valeur . (répondre "O" pour "oui" seulement).
Si la réponse est Non , il vous demande si c'est d'une primitive graphique qu'il s'agit.
Si la réponse est encore "non , il vous demande s'il s'agit d'un primitive à exécution particulière. (le nom de la procédure qui ce charge de son exécution doit alors toujours commencer par le caractère "%")

Remarque: Encore une fois aucune vérification n'est faite par la simulation, c'est à l'utilisateur de vérifier qu'il tape correctement le nom de la primitive.
Pour ajouter une procédure, la méthode est identique,il ne faut cependant pas oublier le fait que toute procédure doit être définie.

[1] Cette simulation n'explicite pas le fonctionnement des procédures. Celles-ci, comme n'importe quelle primitive Logo, sont exécutées globalement sans que soit expliciter leur exécution.

Ajoutons, par exemple, la procédure LONGUEUR qui retourne le nombre de lettres d'un mot.

```
POUR LONGUEUR :L
SI VIDE? :L [RENDS 0]
RENDS SOMME 1 LONGUEUR SP :L
FIN
```

Il suffit ensuite d'appeler AJOUTEPRIM pour la faire mémoriser par la simulation.

si vous désirez sauver vos tranformations , reférez vous à votre MANUEL LOGO.

Modifier la gestion de PP

Vous pouvez facilement modifier la gestion de la pile de paramètres. Il est possible,par exemple,de supprimer la gestion automatique de PP (ce sera alors à vous de gérer celle-ci).En outre vous pouvez parfaitement définir une toute autre manière de gérer PP que celle que vous suggère notre simulation.Pour ce faire vous disposez de trois actions sur la Pile de Paramètres:

DEC.SPP: qui décrémente le sommet de la pile PP;

OTE.SPP: qui ôte le sommet de la pile PP;

DEP.PP: qui attend une donnée et la dépose au sommet de PP.

Supprimer la gestion automatique de PP

La variable AUTOPP, permet de passer directement du mode de gestion automatique de PP au mode ou la pile PP n'est pas gérée par le système. Initialement, AUTOPP est VRAI. Pour supprimer la gestion automatique, il suffit donc de taper:

DONNE "AUTOPP "FAUX

puis de lancer la simulation en tapant DEBUT. En gérant PP à la main, il vous faudra faire (pour interpréter la ligne EC 3) la séquence d'action suivante:

Ordre	Contenu des variables			
	BC	PS	**BE**	PP
TFCS	[3]	[EC]		
DEP.PP 1	[3]	[EC]		[1]
TFCS		[EC 3]		[1]
DEC.SPP		[EC 3]		[0]
TFSE		[EC]	[3]	[0]
TFSE			[EC 3]	[0]
OTE.SPP			[EC 3]	
EXEC				
FIN!				

Les ordres liés à la gestion de PP sont mémorisés, ce qui vous permet d'utiliser REPRENDS. En mode CACHE, il faut utiliser la fonction NPAR pour connaitre le nombre d'arguments attendus par une primitive. Le prédicat PRIMI? est tout aussi indispensable pour savoir le type d'objet que l'on manipule.

Remarque: en mode VISIBLE les utilisateurs ont souvent tendance à se laisser guider par ce qu'ils voient sur l'écran, et oublient de surveiller PP . Le mode CACHE, par contre,introduit des difficultés sérieuses qui ne peuvent se résoudre convenablement si l'on n'est pas attentif à ce qui se passe dans la pile PP.

Pour revenir à la gestion automatique, il suffit de taper: DONNE "AUTOPP "VRAI

Le passage d'une gestion à l'autre est simple, mais il ne peut se faire n'importe quand.Par exemple il ne peut se faire en cours d'interprétation,en tous cas sans risque... Après avoir changé le mode de gestion de PP, il faut donc taper DEBUT pour réinitialiser le système.

Nouvelles gestions de PP

Tout ce qui a été vu jusqu'à présent correspond "grossc modo" au fonctionnement des interprètes LOGO,en particulier ceux implantés sur le TO7 et sur le MO5 (Par ACT). Il est toutefois possible de transformer complétement notre simulation afin de s'approcher d'interprètes LOGO très différents. Voyons comment faire travailler autrement notre machine.

L'action sur la pile PP des transferts TFCS, TFSE, EXEC et RETOUR est commandée respectivement par les procédures TFCS_PP,TFSE_PP, EXEC_PP et RETOUR_PP.

Changer la gestion de PP revient à changer ces procédures. Examinons ce que cela donne sur un exemple:

TFCS

```
POUR TFCS_PP
SI PRIMI? DER :PS [DEP.PP NPARA DER PS]
 [DEC.SPP]
FIN
```

Commentaire: si l'élément à transférer est une primitive: empiler dans PP le nombre d'arguments attendus par cette primitive;si non décrémenter le sommet de PP.

TFSE

```
POUR TFSE_PP
FIN
```

Commentaire : n'a aucune action sur PP.

EXEC

```
POUR EXEC_PP
OTE.SPP
FIN
```

Commentaire: ôte le sommet de la pile PP.

RETOUR

```
POUR RETOUR_PP
DEC.SPP
FIN
```

Commentaire: décrémente le sommet de la pile PP.

La modification que l'on vient d'effectuer ne change pas l'interprétation générale des lignes LOGO.Mais elle élimine un test qui est nécessaire si l'appel de OTE.SPP se fait dans TFSE.

Si vous désirez rajouter des primitives plus complexes (telles que les parenthèses) , il est souvent plus commode d'appeler OTE.SPP dans TFSE.

La série de modifications que nous venons de faire montre bien la grande part de liberté qui est laissée au programmeur dans la définition d'un interpréteur.
Vous restez totalement libre de choisir le type de fonctionnement de l'interpreteur sur lequel et avec lequel vous voulez travailler. Il reste à faire preuve d'un peu d'imagination pour tester d'autres modes de fonctionnement ...

Les différents messages d'erreur

Nous allons passer en revue tous les messages d'erreur que vous pouvez être amenés à rencontrer en utilisant cette simulation.

Nous verrons dans une première partie les messages qui peuvent etre retournés lorsque vous travaillez en mode pas-a-pas, i.e. en choisissant vous-même les différentes actions à effectuer.

La deuxième partie correspond au mode DEMO (ou INTERPRETE) et recense les messages rencontrés avec l'interprétation de lignes LOGO généralement non exécutables. Elle va nous permettre une comparaison entre plusieurs Logo.

Dans la troisième partie, nous verrons certaines erreurs dues à l'utilisation illicite des transferts. On peut rencontrer ces dernières aussi bien en mode interactif qu'en mode d'interprétation automatique.

Un dernier type d'erreurs peut survenir dans le cadre de l'utilisation de ce logiciel. Tous les ordres que vous tapez au clavier sont d'abord interprétés par le LOGO. Si l'ordre que vous envoyez n'est pas compris par celui ci,il vous retourne un message d'erreur. C'est en particulier le cas si vous faites une faute d'orthographe en voulant lancer une des primitives de la simulation. Si vous écrivez par exemple TTCS au lieu de TFCS, Logo vous répondra qu'il ne connait pas TTCS. Il vous suffit de retaper l'ordre pour continuer, cela ne perturbe en rien le déroulement de la simulation.De plus ces messages apparaitrons en lettres majuscules (Alors que tous les messages envoyés par la simulation sont écris en lettres minuscules).

Erreurs en mode interactif

Nous allons faire un rapide tour des primitives de la simulation et des messages qui peuvent apparaître lors de leur utilisation.

TFCS: deux types d'erreurs peuvent se produire:
- si une tentative de transfert est faite alors que BC est vide, un message s'affiche à l'écran:
C'est impossible, il n'y a rien dans BC.

- une tentative de transfert d'une procédure non définie (appelons la TOTO) engendrera un message dû à la simulation:
Je ne connais pas TOTO

TFSE: Une tentative de transfert depuis la pile de stockage alors que celle-ci est vide provoque la même protestation:
C'est impossible, il n'y a rien dans PS.

EXECU: Avec EXECU de nombreux messages peuvent survenir, suivant ce que contient le buffer d'exécution.

- si BE est vide:
C'est impossible, il n'y a rien dans BE.

- si le premier élément du buffer d'exécution BE n'est pas une primitive reconnue par la simulation, l'exécution est refusée. Ainsi, si BE contient [4 5], la commande EXECU retournera le message:
Execution impossible. 4 n'est pas une primitive connue du systeme

On aurait un message similaire si BE contenait VT (qui ne fait pas partie des primitives connues de la simulation).

- s'il y a plusieurs primitives dans BE:
Exécution impossible. Il y a trop de primitives.

- si une primitive n'a pas tous ses arguments, par exemple, si BE contient [SOMME 6]:
Exécution impossible. Pas assez de données pour SOMME

- si une primitive à plus d'arguments que nécessaire,par exemple, si BE contient [EC 3 7 5]:
Exécution impossible. Trop de données pour EC

RETOUR : - si VAL est vide, on ne peut rien ne peut être rendu à lapile PS, ce qui donne:
C'est impossible, il n'y a rien dans VAL.

FINI: - si l'une des diverses piles PS, PP ou un des buffers BC, BE ou VAL n'est pas vide, le traitement n'est certainement pas terminé, de toutes façons la procédure suivie n'a pas été entièrement correcte. On obtient le message :
Il reste des choses à traiter.

- si par contre, tout s'est bien déroulé on obtient simplement:
O.K.

Erreurs en mode d'interprétation automatique

Nous allons voir comment comprendre les messages d'erreur traditionnels de LOGO à l'aide de cette simulation.

Il faut partir du fait que les divers messages d'erreurs se produisent à des moments précis durant l'interprétation d'une ligne.On va voir que cela va nous nous amener à revoir les différentes actions de la simulation sous un nouvel éclairage.

Que fait un message d'erreur? Il interrompt l'exécution de la procédure en cours et renvoit au niveau supérieur (c'est ici qu'intervient la procédure AR (pour ARet) son action est précisée dans l'Annexe Technique du guide d'utilisation.)

Le transfert (TFCS) enlève,comme on l'a vu,le premier élément du buffer clavier pour le mettre au sommet de la pile de stockage. La mise à jour correspondante de la pile de paramètres dépend de la nature de l'objet transféré. En effet, durant le transfert, l'objet est analysé par le système. Ce travail est effectué par la procédure CONT, qui va déterminer quel est le type de l'objet manipulé avant de le rendre à la pile de stockage.

Les objets n'ayant pas pour premier élément un caractère spécial (:) ne peuvent être que: des nombres; des listes; des primitives ou des procédures. [1]

Si l'objet n'appartient pas à l'une de ces catégories , cela signifie que l'objet n'a pas été défini.

Ainsi, si on a utilisé le mot TOTO sans le définir préalablement on obtient:

je ne connais pas TOTO

[1] Rappelons qu'un procédure est traitée dans cette simulation de la même manière qu'une primitive: elle y est considérée globalement sans que son fonctionnement soit explicité.

Si l'objet considéré est précédé du symbole ":" (par exemple :TOTO), c'est alors un nom dont on va aller chercher la valeur. La procédure CONT renvoie alors:

Je cherche l'objet associé àTOTO

Si la recherche n'aboutit pas, on obtient finalement:

TOTO n'est associé à aucun objet.

ce qui arrête la simulation .

Ce dernier message peut aussi être généré au moment de l'exécution de la primitive CHOSE (nommée OBJET dans la simulation). On peut aussi et de la même manière, le rencontrer en mode interactif.

Si on essaye la ligne "EC", le système ne pourra pas l'effectuer. On obtient:

Pas assez de données pour EC

En effet, la consultation (dans AUTO)du sommet de la pile de paramétres PP (qui est égal à 1) invite à continuer le transfert du buffer clavier vers la pile de stockage (TFCS), ce qui échoue puisque BC est vide.

Un autre message très classique se produit si on tente d'exécuter une ligne telle que:

"EC 2 1"

Une telle ligne provoque l'exécution correcte de "EC 2", puis une tentative d'exécution de "1".Mais comme PP ne peut être décrémentée lors de TFCS-PP (puisque PP est vide) on obtient donc:

Ne sais que faire de 1

Ce test est réalisé dans DEC.SPP car il est aussi valable pour (par exemple) SOMME 3 4 qui retourne 7 à PS alors que PP est vide.

Certains messages interviennent quand LOGO essaye d'exécuter certaines actions alors qu'elles sont...inexécutables. Dans ce cas, les messages de la simulation sont ceux de LOGO

En voici quelques exemples:
SOMME "TOTO 4

Dans ce cas , la simulation ne rencontre pas de problèmes de transfert et demande à LOGO d'exécuter cette phrase . Ce dernier ne pouvant pas le faire émet alors le jolie message suivant (en majuscule bien sûr) :

DANS EX, SOMME N'AIME PAS TOTO

Traitons, maintenant la ligne EC AV 100.

Dans tous les cas AV 100 est exécuté, puis l'interprétation s'interrompt et l'on récupère le message suivants:

AV n'a pas produit d'objet pour EC

Ici , La procédure EXECU conserve dans MEM le nom de la derniere primitive exécutée. S'il n'y a pas d'objet dans VAL après l'exécution et si la pile PP n'est pas vide, la procédure PPRIM va chercher dans PS la dernière primitive stockée puis génère le message.

Examinons maintenant les derniers messages d'erreur que vous risquez de rencontrer. On a vu que si on cherche à transférer un objet à partir d'une pile ou d'un buffer vide, on obtient le message:

C'est impossible, il n'y a rien dans ....

Si on essaye de savoir le nombre d'arguments d'un objet quelconque (qui n'est pas une primitive) ou d'une primitive (disons TOTO) alors que celle-ci n'est pas connue du système,on obtient:

je ne connais pas TOTO

Les messages suivants ne peuvent pas, à priori, être rencontrés en mode INTERPRETE. Cependant, des actions illicites précédant un recours à cette procedure peuvent conduire à les générer.

Par exemple, essayons de traiter la ligne EC 1.

Si l'on effectue successivement TFCS puis TFSE, la simulation se trouve dans l'état suivant:

Ordre	**Contenu des variables**			
	BC	PS	PP	BE
TFCS	[1]	[EC]	[1]	
TFSE	[1]		[1]	[EC]

Dans cette situation, le système est incapable d'exécuter correctement la ligne.L'appel d'INTERPRETE conduit donc au message:

Pas assez de donnees pour EC

Etant donné l'état où nous avons laissé le système, avant l'appel d'INTERPRETE, notre interpréteur est incapable de reconnaitre que quelque chose ne va pas: BE ayant une primitive comme premier élément, INTERPRETE appelle donc aussitôt EXEC...

Un autre exemple d'une ligne inexécutable est simplement la ligne:

3 4.

Si on tape TFCS, puis INTERPRETE, on aura le message:

Ne sais que faire de 4

Mais si on a, au préalable, empilé 2 par exemple dans PP, à l'appel de INTERPRETE, PP n'étant pas vide, le système "suppose" qu'une primitive a déjà été empilée dans PS. Il va alors transférer un à un les éléments de BC jusqu'à ce que le sommet de PP soit égal à 0. Or après avoir transféré le 4, BC est vide, et le système va alors renvoyer un message (étrange mais cohérent) signalant que cette primitive n'a pas assez d'arguments, ce qui donne:

Pas assez de données pour une primitive absente

Dans tous les cas, si le recours à la procédure INTERPRETE conduit à des messages non standards, c'est qu'on a dû au préalable effectuer certaines actions qui ont laissé la simulation dans une situation "anormale".

Pour relancer une interprétation correcte, il faut taper REPRENDS afin de réutiliser la même ligne, puis taper sur la touche S pour ne pas refaire les actions précédentes, enfin il faut taper INTERPRETE .

Pour changer les messages d'erreur

Les messages d'erreur sont classés par leur ordre de numérotation dans la variable ERR.

Pour intervenir facilement sur cette variable , il suffit de passer par l'éditeur. (ED ["ERR]).Mais ATTENTION bien qu'il soit facile d'adapter les messages d'erreur à une situation particulière, il est recommandé de bien se représenter comment et quand ces messages vont apparaître (afin que leur signification soit clair).

Eviter en particulier l'usage :

- des majuscules (pour différencier vos messages et ceux du LOGO)

- des "." en fin de ligne.

Remarque : vous pouvez ,bien sûr,ajouter d'autres messages d'erreur à la liste prévue initialement. Ceci n'a de sens que si vous intervenez suffisamment dans la simulation pour que le type de message que vous introduisez ait une certaine généralité (il faut aussi que vous fassiez figurer le numéro de vos messages d'erreur dans ERR). Reportez vous pour cela à l'Annexe Technique.

Quelques pistes d'utilisation pédagogique

Dans ce court chapitre, il ne s'agit pas de donner exhaustivement toutes les possibilités d'utilisation de cette simulation, mais de proposer quelques exemples permettant de guider efficacement les formateurs.Il est important que ces derniers puissent s'approprier complètement ce logiciel, et qu'ils le "détournent" suivant leurs besoins.

Une progression ?

L'architecture de ce guide d'utilisation donne déjà une première approche de la facon dont on peut proposer ce logiciel dans un cadre de formation. En gros, les différentes étapes pourraient être les suivantes:

- mode DEMO sur des lignes utilisant les primitives arithmétiques;

- mode pas-à-pas sur le même type de lignes (choisies par l'utilisateur ou données par l'option EXO);

- comparaison de différentes interprétations à priori possibles d'une même suite d'ordres;

- utilisation du mode CACHE pour s'initier à la généralisation de divers modes d'interprétation;

- enrichir les lignes en travaillant avec d'autres primitives (LISCAR, les mots, les listes, les booléens, les variables);

- refléchir sur les diverses erreurs rencontrées en analysant bien leur nature et le moment précis ou elles interviennent. Il serait alors intéressant de tenter d'interpréter des lignes inexécutables et de changer les messages d'erreur;

- ajouter de nouvelles primitives et les essayer dans la simulation;

L'aboutissement de ce type de travail peut conduire à réécrire la procédure INTERPRETE [1] puis à la tester sur des lignes différentes.

De manière complémentaire, on peut supprimer la gestion automatique de PP, puis changer le fonctionnement général de cette pile en réécrivant les procédures TFCS_PP, TFSE_PP, EXEC_PP et RETOUR_PP.

[1] En fait la procédure AUTO de la simulation.

Remarques générales

Il est clair que suivant les compétences des utilisateurs, ce logiciel ne devra pas être vu de la même manière. Par exemple le mode DEMO peut très bien ne jamais être montré , si dès le début on se donne pour tâche d'essayer de construire une image ou un modèle d'interpréteur de ce type (ce qui suppose une bonne connaissance du langage LOGO).

Si l'important est seulement de donner une idée du fonctionnement d'un interpréteur, sans trop chercher à approfondir les détails de son fonctionnement, alors au contraire le mode DEMO avec la visualisation de la machine est un bon outil de base.

Annexe technique

Cette annexe est destinée aux utilisateurs désireux de s'approprier plus complètement la simulation. Elle va nous permettre de préciser comment sont fabriquées certaines des fonctions essentielles du logiciel:
TFCS,TFSE,EXECU etc.

La procédure CONT est appelée par TFCS, et contrôle le type de données que l'on essaye de transférer de BC à PS. De plus, si un mot est précédé du caractère spécial (:), elle va retourner sa valeur.

EXEC? prend une liste comme donnée, et regarde si cette dernière est à priori exécutable (c'est-à-dire si elle est non vide et si son premier élément est une primitive connue du système).

TESTEXEC est une commande qui vérifie si le contenu de BE (supposé non vide) peut être exécuté,elle vérifie que:

-le premier mot est bien une primitive

-c'est la seule primitive présente

-le nombre d'arguments est correcte

L'interpretation automatique

La procédure DEMO ne fait que lancer INTERPRETE apres DEBUT.

La primitive INTERPRETE efféctue deux actions prioritaires suivant l'état dans lequel se trouve la simulation:

- si VAL est non vide, un RETOUR est envoyé

- si BE est exécutable (EXEC? :BE retourne VRAI), EXECU est envoyé, suivi éventuellement d'un RETOUR.

Après cela, INTERPRETE appelle la procédure récursive AUTO qui s'occupe du reste... C'est en fait elle qui assure l'interprétation dans notre machine ,mais attention elle ne fait que cela .Ainsi, quand on lance AUTO aucun bilan n'est établi sur l'état courant, ce qui peut amener certains dysfonctionnements.

Empècher la procédure AUTO de partir sur une fausse piste est un bon exercice,nous le conseillons au lecteur courageux.

Les trois principaux messages d'erreur dans l'interprétation correspondent à trois procédures:

QUEFAIR -----> Ne sais que faire de ...

MANQUE -----> Pas assez de données pour ...

PASDOBJ -----> ...n'a pas produit d'objet pour ...

Elle sont appelées respctivement par DEC.SPP, AUTO et AUTO.

Le problème de la sonorisation

Si vous avez l'oreille sensible vous pouvez réduire les agressions sonores de votre ordinateur en modifiant les procédures SONNE,
(elle intervient à chaque exécution) et ERR, qui ponctue les messages d'erreur.

MANUEL DE REFERENCE

Introduction

Note sur la lecture du manuel de référence

Le chapitre "Description de la représentation" récapitule brièvement les diverses fonctions des piles et des buffers ainsi que les ordres que l'on peut donner dans le cadre de la simulation.

Avant de le lire, il est conseillé d'avoir déjà fait fonctionner notre simulation, au moins en mode DEMO.

Le reste de ce manuel doit être considéré comme une sorte de "dictionnaire" auquel vous pouvez recourir lorsque vous voulez vous remettre en mémoire la syntaxe ou le fonctionnement d'un ordre.

Rappelons que le modèle décrit ici est une représentation minimale du décodage des ordres LOGO.

Sa caractéristique principale est de permettre un accès intuitif (visuel) à ces systèmes formels que l'on appelle des interpréteur.Rappelons aussi les quelques contraintes qui régisssent globalement notre interpréteur.

(a) il faut éviter l'interprétation d'expressions concernant le graphique .Notre interpréteur ayant lui même un statut graphique on comprend que la visualisation de la simulation n'est plus possible lors de l'exécution d'une primitive graphique.

(b) dans la version de base qui vous est proposée l'interprétation de procédures ou de listes n'est pas detaillée; il est toutefois possible d'étendre le fonctionnement de la simulation à ces objets si on le souhaite .C'est un exellent exercice.

C'est d'ailleurs en songeant à des développements de ce genre que ce logiciel a été conçu. Les ordres comme SI, REPETE, EXECUTE, ne sont pas non plus traités.
Mais de même que pour les procédures, il est possible d'étendre la simulation afin de les y inclures

(c) pour distinguer les objets (variables ou primitives) qui peuvent être à la fois utilisés par notre simulation et par LOGO, nous avons dû introduire un vocabulaire spécifique à notre simulation.Par exemple dans le cas des variables nous avons introduit

ASSOCIE au lieu de DONNE
et OBJET au lieu de CHOSE.

Description de la simulation

Structure du simulateur

Notre interpréteur peut être compris comme une machine sur laquelle on peut agir et qui visualise le résultat de ses actions.

Les "pièces" de la machine sont: des zones mémoire où sont stockées les informations.Elles sont au nombre de cinq :

(1) Un Buffer Clavier (BC) autrement dit une zone mémoire, où sont stockées les informations en provenance du clavier.

(2) une Pile de Stockage (PS), où sont stockées les instructions en cours de traitement, et leurs arguments.

(3) une Pile de Paramètres (PP),son sommet indique le nombre de paramètres de la dernière instruction devant encore être empilée dans PS.

(4) Un Buffer d'Exécution (BE) zone mémoire dont le contenu est transmis à LOGO au moment de l'exécution.

(5) une zone mémoire (VAL) qui contient le résultat de l'exécution de BE quand celle-ci produit une valeur.

Panorama de DEMO

Voici une macro-description du programme de démonstration (DEMO).Notre objectif ici est de donner une vue d'ensemble de notre petite machine à interpréter.Bien entendu il vous sera beaucoup plus facile de comprendre les indications qui suivent si vous avez vu tourner DEMO au moins une fois (pour cela il suffit de taper DEMO).

- Une suite d'ordres est stockée dans le buffer clavier.

- Ces informations sont transférées une par une dans la pile de stockage, en mettant à chaque fois à jour la pile de paramètres. Cette mise à jour est auto- matique.

- Si l'objet transféré est une primitive on empile dans PP le nombre d'arguments qu'elle attend. Sinon on décrémente le sommet de PP.

Ainsi, le sommet de la pile de paramètres indique si la dernière primitive empilée dans la pile de stockage attend encore ou non des arguments.

- Quand le sommet de pile de paramètres vaut 0, c'est donc que la dernière instruction empilée dans la pile de stockage n'attend plus d'argument. On transfère alors tout ce qui est dans PS dans le buffer d'exécution jusqu'à ce que l'on aboutisse à une primitive.

- Ce qui est dans le buffer est alors exécuté par l'interpréteur Logo. Quand celui-ci retourne une valeur, elle est stockée dans la case mémoire VAL, puis le contenu de celle-ci est ensuite transféré dans la pile de stockage.

Les procédures utilisateur

La procédure d'initialisation

DEBUT

Action:

Initialise les piles et les buffers .L' utilisateur a le choix soit de taper une suite d'ordres à traiter, soit de laisser le système lui fournir une ligne à exécuter, en tapant EXO. Cette suite d'ordres est alors mise dans BC. A partir de là, le système mémorise toutes les actions effectuées.

Les procédures de transfert

TFCS

Action:

Transfère le premier élément du buffer clavier dans la pile de stockage. Après cet ordre, le sommet de PS est l'ancien premier élément de BC,tandis que BC a perdu son premier élément. La pile de paramètres, PP, est automatiquement gérée .

Si le premier élément de BC est une primitive, le nombre d'arguments attendus par celle-ci est empilé dans PP, sinon le sommet de PP est décrémenté.

Exemple:

Si BC vaut [EC SOMME 2 3] , PS vaut [ ] et PP vaut [ ], après l'ordre TFCS, ils valent respectivement : [SOMME 2 3] , [EC] et [1].

TFSE

Action:

Transfère le sommet de la pile de stockage dans le buffer d'exécution . Après cet ordre, BE a pour premier élément le sommet de PS, et PS perd son sommet.

Si l'objet transfèré est une primitive, le sommet de PP est ôté.

Exemple:

Si PS vaut [EC SOMME 2] et BE vaut [3], après l'ordre TFSE, PS vaut [EC SOMME] et BE vaut [2 3].

EXEC

Action:

transfère le contenu du buffer d'exécution vers l'exécuteur Logo qui traite alors les données fournies et rend, s'il y a lieu, un résultat.

Si l'utilisateur a employé des primitives graphiques, celles-ci sont mémorisées pour être exécutées à la fin de ce programme.

Si l'exécution retourne une valeur, celle-ci est stockée dans VAL.

Si le premier élément de BE n'est pas une primitive ou si BE contient plus d'une primitive, l'exécution est refusée. Après cet ordre, le buffer d'exécution est vide.

Exemple:

Si BE vaut [EC 50], l'ordre EXEC donne à l'exécuteur Logo la liste [EC 50] à exécuter, puis vide BE.

Si BE vaut [SOMME 1 2], l'ordre EXEC donne à l'exécuteur Logo la liste [SOMME 1 2] à exécuter, puis vide BE, et enfin met la valeur 3 dans VAL.

RETOUR

Action:

Transfère le contenu de la case mémoire VAL dans la pile de stockage et décrémente le sommet de PP.

Après cet ordre, le sommet de la pile de stockage vaut le contenu de VAL, puis VAL est vidée.

Exemple:

Si PS vaut [EC SOMME 1], PP vaut [1 1] et VAL vaut [3], après l'ordre RETOUR, PS vaut [EC SOMME 1 3], PP vaut [1 0] et VAL vaut [ ].

Les procédures système

DESSIN

Action:

Réaffiche le graphisme des piles et des buffers.

Cette procédure charge le fichier "INTERPRE.DES" Qui se trouve au début de la cassette et il vous sera demandé à chaque fois de ramener la cassette au départ .

Si vous ne désirez pas charger le dessin tapez "I" .

ANNULE

Action:

Annule la dernière commande effectuée par l'utilisateur, remet la visualisation graphique à jour. Le système refait alors toute la suite d'actions, sans les visualiser, sauf la dernière.

REPRENDS

Action:

Reprends en pas à pas l'exécution des actions effectuées par l'utilisateur ou le système. Pour lancer chaque action, l'utilisateur doit taper une touche quelconque. La touche "S" permet d'interrompre le déroulement des opérations en laissant le système dans l'état courant.

DEMO

Action:

Traite automatiquement une suite d'ordres soit tapée par l'utilisateur, soit choisie parmis les nombreux fichiers d'exercices (en tapant EXO) .

Cette procédure utilise DEBUT.

INTERPRETE

Action:

Transfère le controle de la simulation au système qui va tenter de finir l'interprétation de la ligne en cours.

Le succès de cette interprétation depend, bien sûr, de l'état du système au moment ou vous avez appelé INTERPRETE.

FINI

Action:

Indique à la simulation que l'utilisateur a fini d'agir, l'interpréteur vérifie alors si l'état des piles et des buffers est cohérent. Si l'utilisateur a employé des primitives graphiques, on lui demande s'il souhaite voir exécuter ces primitives. Si oui, l'écran graphique est "nettoyé" pour permettre cette exécution.

CACHE

Action:
Après cet ordre, seul le contenu de PP est lisible, l'affichage des contenus des autres piles et buffers se fait à l'aide de pavés.

Exemple:
Si BC vaut [SOMME LISCAR 2], PS vaut [EC] et PP vaut [1],
après CACHE la visualisation de BC sera [X X X], celle de PS sera [X], et celle de PP sera [1].

VISIBLE

Action:
Annule l'effet de l'ordre CACHE et remet la visualisation graphique à jour.

Renseignements sur les primitives

PRIMI? objet

Action:
Retourne VRAI si l'objet est une primitive connue de notre modèle, et FAUX sinon.

Exemple:
EC PRIMI? "SOMME affiche le message VRAI,
EC PRIMI? "REPETE affiche le message FAUX.

NPAR objet

Action:
Retourne (si l'objet est une primitive connue du système) le nombre de paramètres utilisés par celle ci. Sinon envoie un message d'erreur à l'utilisateur.

Exemple:
NPAR "LISCAR retourne 0,
NPAR "PRODUIT retourne 2.
NPAR "GRBLWX retourne le message
Je ne connais pas GRBLWX

PRIM

Action:
Retourne la liste des toutes les primitives utilisables de la simulation. Cette liste peut être completée par l'utilisateur [1]

?

Action:
Appelle le menu d'aide-mémoire des primitives de la simulation (par l'intermèdiaire de PRIM) .

[1] Voir le chapitre "ajouter des primitives" dans le guide d'utilisation.
[2] Voir AUTOPP dans le chapitre "Les variables utilisateur" du manuel de référence.

Gestion manuelle de PP [2]

DEP.PP nombre

Action:
Dépose le nombre spécifié comme argument au sommet de la Pile des Paramètres .

OTE.SPP

Action:
Ote le sommet de la Pile des Paramètres .

DEC.SPP

Action:
Décrémente le sommet de la Pile des Paramètres .

Les variables utilisateur

Voir

Syntaxe:
DONNE VOIR "VRAI ou "FAUX

Action:
Par défaut,VOIR est vrai: cela correspond à la procédure VISIBLE. Dans le cas contraire, la visualisation s'efféctue en mode CACHE.

PG

PG contient la liste des primitives graphiques connues du système: AV, TD, TG, RE, BC, LC, FCC, FCF,VE.

P0

P0 contient la liste des primitives qui ne retournent rien:
EC, ASSOCIE, TAPE.

P1

P1 contient la liste des primitives qui retournent une valeur:
SOMME,DIFF, PROD,DIV, RESTE,QUOT,PREM,DER, HASARD, DIFF , OBJET, SP, SD, MOT, NON, OU,ET, EGAL?, PH.

P%

P% contient les primitives du Logo dont l'exécution est spécifique dans cette simulation: LL,LISCAR.

[1] Pour modifier l'une des variables P0, P1, PG ou P%, voir le chapitre "Ajouter des primitives" dans le guide d"utilisation

ETAT

Dans la variable ETAT sont mémorisées toutes les actions effectuées depuis le debut d'une exécution. Le système s'en sert en particulier lors d'un ANNULE ou d'un REPRENDS.

PL

PL mémorise la liste des valeurs entrées par LL ou LISCAR. Elle est utilisée lors de l'exécution de REPRENDS

MEM

La variable MEM est gérée par la procédure EXEC. Elle mémorise le nom de la derniere primitive exécutée par la simulation, en vue d'afficher des messages d'erreur précis. [2]

AUTOPP

Syntaxe:
DONNE AUTOPP "VRAI ou "FAUX

Action:
La variable AUTOPP est un prédicat qui spécifie le mode de gestion de PP:

- si AUTOPP est VRAI,(par défaut) PP est géré automatiquement par le système,
- si AUTOPP est FAUX, PP n'est pas géré par le système en pas-à-pas, et les ordres donnés à PP sont stockés dans la variable ETAT.

ERR

ERR contient la liste des différents message d'erreur.

MSG

MSG contient la liste des message de l'interpréteur.

[2] Voir dans le chapitre sur les différents messages d'erreurs (Guide d'utilisation) le paragraphe concernant l'ordre RETOUR.

glossaire des primitives de l'interpréteur

1 DEBUT: initialise notre "machine à interpréter" et lance la simulation

2 TFCS: TransFère un objet du buffer Clavier dans la pile de Stockage

3 TFSE: TransFère un objet de la pile de Stockage dans le buffer d'Exécution

4 EXECU : EXECUte le contenu du buffer d'exécution

5 RETOUR: RETOURne la valeur contenue dans VAL à la pile de stockage

6 FINI : déclare que l'interprétation de la ligne introduite au début est FINIE

7 INTERPRETE : demande au système de finir d'INTERPRETER la ligne introduite au début

8 PRIMI? : teste si une PRIMI?tive est connue par le système

9 PRIM: retourne la liste des PRIMitives connues par le système

10 NPAR: retourne le Nombre de PARamètres utilisés par une primitive ou une procédure connue (8841 = no value).

11 ANNULE: annule la dernière action effectuée;

12 REPRENDS: refait une séquence d'instructions à partir du début

13 DESSIN : reconstitue le graphisme de la simulation

14 ? : appelle le récapitulatif aide-mémoire des primitives de ce programme

15 CACHE: Cache à l'utilisateur les contenus de BC, PS, BE

16 VISIBLE: est L'inverse de CACHE.C'est le mode ordinaire (par défaut)

17 DEP.PP: DEPose un élément au sommet de la pile PP

18 OTE.SPP: OTE le Sommet de pile PP

19 DEC.SPP: DECrémente le Sommet de pile PP

Imprimerie Hérissey - Évreux - N° 38256 — D.L. Oct. 1985