PATRIMOINE SCIENTIFIQUE
TECHNIQUE CONTEMPORAIN

PRÉSERVER
SAUVEGARDER
VALORISER

LANGAGE PROLISP

FICHE N° 6513

Période de fabrication : -
Fabricant : Office National d'Etudes et de Recherches Aérospatiales - ONERA
Domaines : Informatique et Communication
Sous-domaines : Logiciel, Ordinateurs, Langage
Organisme : Office national d'Etudes et de Recherches Aérospatiales - ONERA
Ville : Toulouse
Modèle :
Matériaux :

Description

PROLISP est un langage de programmation interprété, c'est à dire qu'un programme, appelé interpréteur est chargé de l'exécution des phrases rédigées dans ce langage . Ce langage développé au début des années 1980 par l'Office national d'Etudes et de Recherches Aérospatiales - ONERA, combine deux des grands langages des années 1970, utilisés dans des applications d'Intelligence Artificielle : LISP et PROLOG. Le langage LISP spécifié en 1958 et développé dans les années 60 a montré ses capacités dans la programmation dite symbolique que l'on rencontre alors dans le développement de systèmes experts et les programmes de jeux notamment. Les deux langages prédominant à cette époque, Fortran et Cobol étaient plus adaptés à aux traitements numériques, scientifiques ou de gestion. Quant au langage PROLOG apparu dans les années 1970, il a introduit un nouveau style de programmation, dit de programmation logique, dans lequel le programmeur spécifie des faits et des règles sur des objets et leurs relations, pour poser des questions sur ces objets et relations; un moteur d'inférence, câblé dans l'interpréteur Prolog, produit alors les réponses. Les avantages et inconvénients de LISP et de PROLOG semblant largement complémentaires, plusieurs chercheurs ont cherché à associer ces deux langages dans un environnement commun : PROLISP développé au CERT est une des toutes premières réalisations françaises d'intégration de PROLOG dans un environnement LISP. L'interprète PROLISP est un ensemble de fonctions LISP, offrant ainsi au programmeur PROLISP des ponts entre les deux langages, comme par exemple la possibilité dans la définition de relations d'appel à des fonctions LISP dont le résultat, à l'exécution, sera associé à une variable logique. Ce type d'association permet d'exploiter au mieux la spécificité des deux langages.

Utilisation

Le langage PROLISP a été très utilisé au Département d'Études et de Recherches en Informatique - DERI, ainsi qu'au Département d'Automatique du CERT - DERA, dans le cadre de travaux de doctorants pour la réalisation de prototypes; dans le cadre de projets de recherche internes au Département pour le prototypage de diverses; dans les stratégies de résolution (interrogations dans des bases de connaissances), pour le couplage avec une base de données relationnelle, pour l'analyse d'exigences Génie Logiciel), pour la description et l'analyse de modélisation conceptuelle de divers systèmes d'informations, etc..; au Département d'Automatique pour des applications dans le domaine de la robotique, planification, etc

CERT

office national d'études
et de recherches aérospatiales

centre d'études et de recherches
de l'école nationale supérieure
de l'aéronautique et de l'espace à toulouse

Rapport final n° 1/3617/DERI

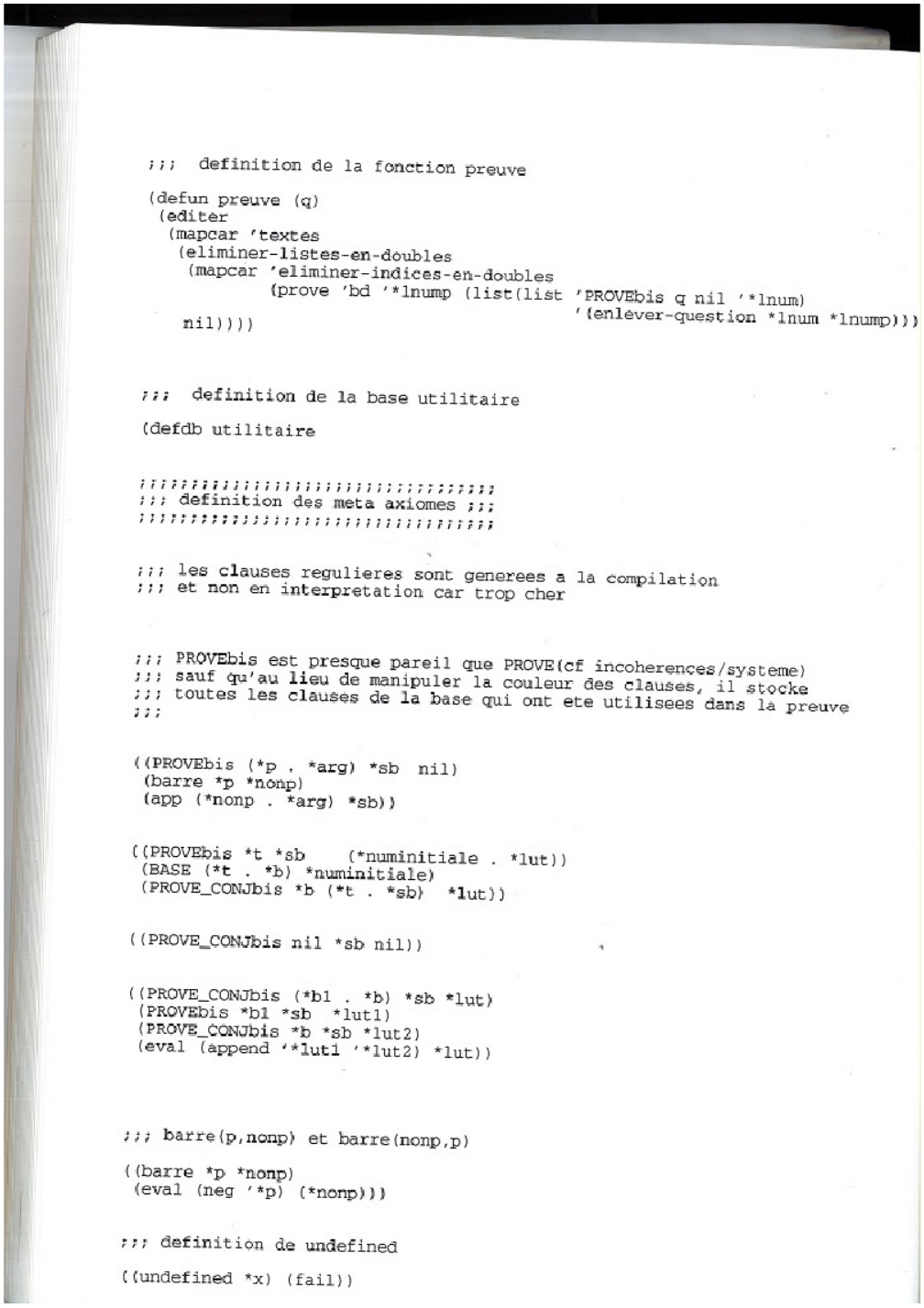
"Prolisp, manuel d'utilisation"

Convention ONERA n° 3617

Septembre 1984

département d'études et de recherches en informatique

2, avenue édouard belin - bp 4025 - 31055 toulouse cedex - tél. (61) 55.71.11 - télex 521 596 F onecert



```
;;;  definition de la fonction preuve

(defun preuve (q)
 (editer
  (mapcar 'textes
   (eliminer-listes-en-doubles
    (mapcar 'eliminer-indices-en-doubles
            (prove 'bd '*lnump (list(list 'PROVEbis q nil '*lnum)
                                          '(enlever-question *lnum *lnump)))
    nil))))


;;;  definition de la base utilitaire

(defdb utilitaire


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; definition des meta axiomes ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;; les clauses regulieres sont generees a la compilation
;;; et non en interpretation car trop cher


;;; PROVEbis est presque pareil que PROVE(cf incoherences/systeme)
;;; sauf qu'au lieu de manipuler la couleur des clauses, il stocke
;;; toutes les clauses de la base qui ont ete utilisees dans la preuve
;;;


((PROVEbis (*p . *arg) *sb  nil)
 (barre *p *nonp)
 (app (*nonp . *arg) *sb))


((PROVEbis *t *sb    (*numinitiale . *lut))
 (BASE (*t . *b) *numinitiale)
 (PROVE_CONJbis *b (*t . *sb)  *lut))


((PROVE_CONJbis nil *sb nil))


((PROVE_CONJbis (*b1 . *b) *sb *lut)
 (PROVEbis *b1 *sb  *lut1)
 (PROVE_CONJbis *b *sb *lut2)
 (eval (append '*lut1 '*lut2) *lut))



;;; barre(p,nonp) et barre(nonp,p)

((barre *p *nonp)
 (eval (neg '*p) (*nonp)))


;;; definition de undefined

((undefined *x) (fail))
```