UPMC/master/info/APS-4I503
Un autre langage
Examen réparti 1

P. Manoury

mars 2015

On considère un noyau impératif classique auquel ou ajoute des *pointeurs*.

1 Grammaire

Lexique

- Les symboles réservés sont : `( ) : ; = < + - * / :=`
- Mes mots clés sont : `ref unref case else loop while int bool true false null begin end var`
- Les constantes numériques sont num (les nombres sont des *flottants*)
- Les identificateurs sont ident

Syntaxe

PROG	::=	`begin` CMDS `end`
CMDS	::=	ε
	|	STAT `;` CMDS
	|	DEC `;` CMDS
STAT	::=	LHS `:=` RHS
	|	`loop` CMDS `while` EXP
	|	`case` EXPR `=` CLAUSES
CLAUSES	::=	`else` CMDS
	|	GUARD `:` CMDS CLAUSES
GUARD	::=	EXPR
DEC	::=	`var` ident `:` TYPEXPR
LHS	::=	ident
	|	`unref` LHS
RHS	::=	EXPR
	|	`ref` RHS
TYPEXPR	::=	`bool`
	|	`int`
	|	`ref` TYPEXPR

EXP	::=	ident
	|	`unref` ident
	|	`true`
	|	`false`
	|	`not` EXPR
	|	EXPR `and` EXPR
	|	EXPR `or` EXPR
	|	EXPR `=` EXPR
	|	EXPR `<` EXPR
	|	`num`
	|	EXPR `+` EXPR
	|	EXPR `-` EXPR
	|	EXPR `*` EXPR
	|	EXPR `/` EXPR
	|	`(` EXPR `)`

2 À faire

Typage Le langage de type est TYPEXPR augmenté de `void`. Les types des expressions seront des TYPEXPR. Le type des instructions sera `void`. La primitive `ref` appliquée à une expression de type τ donne une valeur de type `ref` τ. La primitive `unref` ne s'applique qu'à des valeurs de type `ref` τ. Dans un `case`, toutes les GUARD ont le même type.

Donnez des règles de typage pour ce langage.

Sémantique La primitive `ref` (dans RHS) crée un *pointeur* vers la valeur du RHS qui suit ; elle alloue une adresse qui devient la valeur du LHS correspondant. La primitive `unref` est la primitive d'indirection ; elle donne la valeur contenue à l'adresse donnée par le LHS ou l'EXPR qui la suit. La structure de contrôle CASE est une généralisation de la structure d'alternative : elle exécute le premier bloc de commande dont la GUARD est égale à la valeur de EXPR.

Donnez une sémantique opérationnelle du langage.