RELAZIONE DELLA TERZA PARTE DEL PROGETTO DI LINGUAGGI DI PROGRAMMAZIONE

URBAN DENIS

ZILLE MARCO

MONDINI FRANCESCO

KOSTYUK ROSTYSLAV

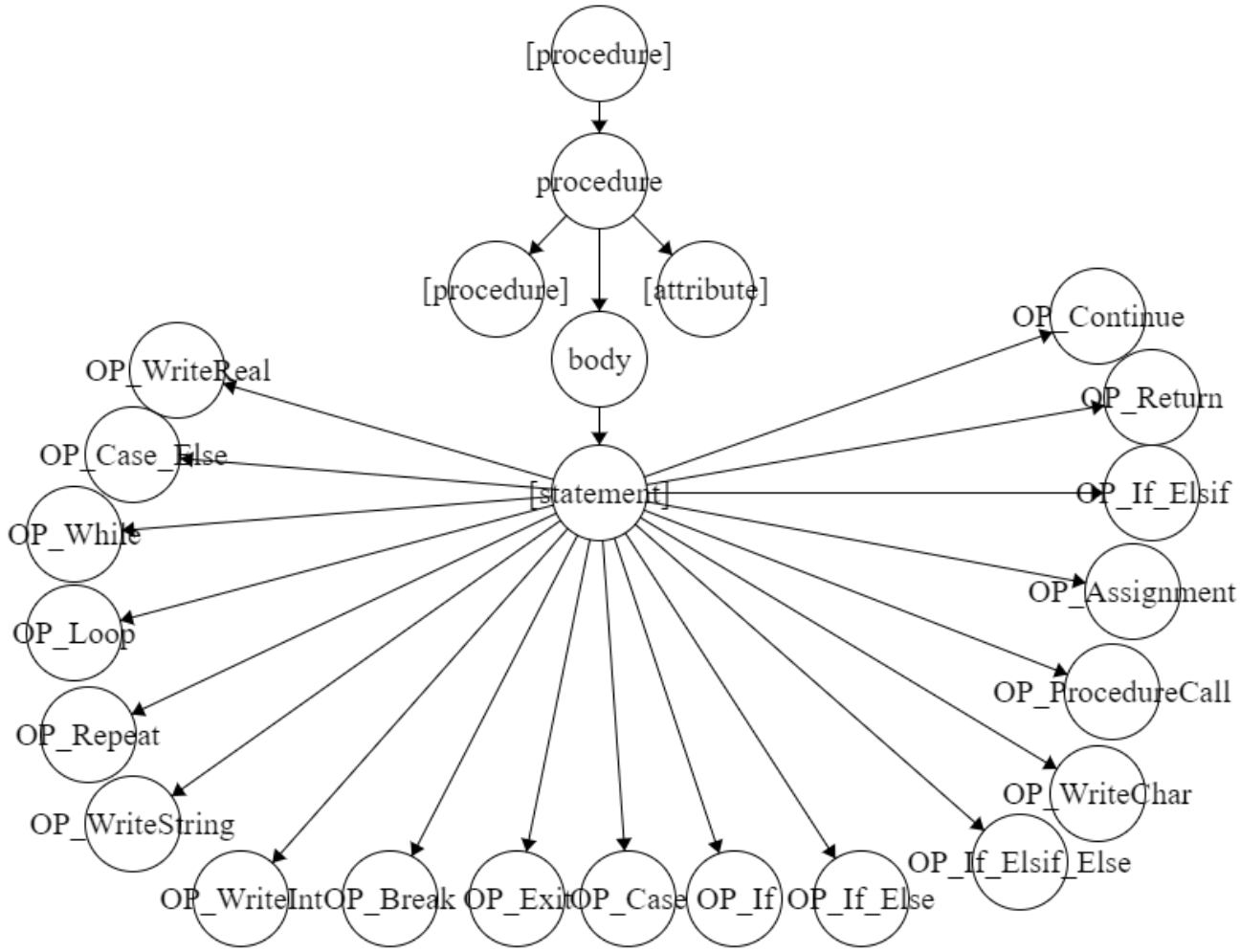
ESERCIZIO 6.

Abbiamo deciso di non tenere presente la direttiva MODULE di Oberon e nemmeno la direttiva TYPE perché non è necessario che l'utente possa definire dei tipi di dato. Inoltre, non abbiamo implementato WITH e SET di Oberon per ragioni pratiche. Abbiamo optato la dichiarazione delle variabili deve avvenire SOLAMENTE all'inizio di una procedura come si nota in questo link http://groups.engin.umd.umich.edu/CIS/course.des/cis400/oberon/ttables.html.

In un file sorgente del linguaggio ci dovrà essere necessariamente una procedura denominata che l'utente deve implementare ed è il punto di ingresso per l'esecuzione del programma che sta scrivendo similarmente a quanto accade in C/C++. Abbiamo, inoltre, deciso di non ammettere dichiarazioni di variabili globali, che potranno essere solamente dichiarate all'inizio delle procedure.

I valori dei tipi base abbiamo deciso di riconoscerli nel seguente modo: i numeri interi come valori compresi tra 0 e 9, i numeri float come ‘numero intero’ , ‘numero intero’, i singoli caratteri come ASCII, le stringhe come {ASCII}, i valori booleani come o True o False ed infine gli identificatori come a...zA...z (numero intero | a...zA...z)*. Per il riconoscimento della dichiarazione delle variabili, abbiamo optato per un riconoscimento nel seguente modo: i numeri interi come “INTEGER”, i numeri float come “REAL”, i caratteri singoli come “CHAR”, le stringhe come “ARRAY OF n CHAR”, i booleani come “BOOLEAN”, gli array come “ARRAY OF n TYPE”, i pointer come “POINTER TO TYPE” e le procedure come “PROCEDURE” (parametri formali). La dichiarazione di variabili avviene come "VAR" identificatore ("," identificatore)* ":" type. Per la dichiarazione delle procedure abbiamo optato per una notazione di questo tipo, "PROCEDURE" nomeProcedura [parametriFormali] [":" type], e, in caso di sequenza di dichiarazioni, ["BEGIN" sequenzaIstruzioni] "END" nomeProcedura. Per i parametri formali è stata scelta una struttura di questo tipo, "(" identificatore ":" type ["," identificatore ":" type]* ")".

Per lo svolgimento di questo esercizio ci siamo appoggiati alle strutture dati nel parser costruita con 8 tipi base: Attribute, Procedure, AttributeType, SimpleType, BasicOperation, Operation, DeclarationType e Declaration. Il primo indica il tipo di attributo, memorizza il valore degli attributi in base al tipo di attributo a cui si riferisce e serve per capire se l’attributo passato è un parametro o sia un valore passato come argomento ad una procedura. L’AttributeType definisce se l’attributo è di tipo semplice, se è un’array di lunghezza intera o indefinita; quest’ultima è usata per la dichiarazione dei parametri formali di una procedura. Il SimpleType definisce il tipo di una striga scelto tra Char, Float, Integer, Boolean, Name, OperationResult o Unknown; l’utilizzo di quest’ultimo implica che l’attributo va calcolato con una variabile o una costante di cui non si conosce ancora il tipo. BasicOperation comprende tutte le operazioni di base tra attributi, dalle operazioni elementari a quelle relazionali. Operation, invece, racchiude tutti gli statements delle funzioni. Procedure definisce una procedura con un nome, delle variabili, degli attributi e restituisce un tipo. DeclarationType definisce se un tipo è variabile, costante, procedura o operazione. Declaration serve per memorizzare la dichiarazione di un attributo variabile o costante, una procedura o un’operazione.



Grammatica Astratta 1

Sono presenti 5 file di test di cui il file Test4 contiene degli errori, mentre Test5 ha tutto.

Purtroppo, per mancanza di tempo, non siamo riusciti a completare il prettyprinter e il typechecker.