Progetto di Linguaggi di Programmazione

Anno Accademico 2022/2023

In questo documento trovate le informazioni necessarie sul progetto. Vi ricordo che il progetto deve essere fatto da gruppi di 3 - 4 persone. Ogni gruppo ha una o un *responsabile* che è anche colei/colui che deve comunicarmi, via email, i nomi dei componenti del gruppo. Il termine per la formazione dei gruppi sarà il **6 gennaio 2023**. I gruppi costituiti successivamente non hanno diritto ad alcun bonus e lo stesso vale per quei gruppi in cui la composizione viene modificata. Per chiarimenti scrivetemi.

Tale progetto non è vincolante, ciascun gruppo può proporre un progetto diverso che deve riguardare almeno l'interprete, il lexer ed il parser di un linguaggio che ciascun gruppo può proporre. Il progetto proposto deve essere prima sottoposto ad approvazione e solo in questo caso può sostituire il progetto dato.

Consegna del progetto

Il o la responsabile deve creare un repository privato su GitHub, e invitare come collaboratori tutti i componenti del gruppo, e i docenti (`gmpinna` e `bitbart`). Per poter avere il bonus il progetto va consegnato entro il **15 febbraio 2023**.

Il repository deve contenere tutti i file sorgenti del vostro progetto, generato con `dune init proj lip22`. Il progetto deve essere compilabile con `dune build` senza generare errori o warning. Inoltre, il progetto deve usare i files presenti nel repository:

`https://github.com/informatica-unica/lip/tree/main/imp/lip22`

Questi files *non* possono essere modificati, ad eccezione del file `test/tests`.

Il linguaggio **Repeat**

Il linguaggio di programmazione di cui dovrete scrivere l'interprete, il lexer ed il parser è descritto di seguito. Dovrete realizzare la *small step semantics*, mentre in questa descrizione viene data la big step semantics. Il linguaggio ha dichiarazioni di variabili che devono essere di *tipo* intero (pur non avendo né un type checker né si fa inferenza di tipo), di *array*, sempre di tipo intero, e procedure che hanno un solo parametro formale che può essere passato per valore o riferimento. Le dichiarazioni di procedura possono essere fatte solo nel blocco piú esterno, quindi sono dichiarazioni globali. Il linguaggio ha *side effects*.

Dichiarazioni: Per le dichiarazioni di variabile e array abbiamo la seguente grammatica:

$$dv \quad ::= \quad \mathsf{nullvar} \mid dv_1; dv_2 \mid \mathsf{var}(X) \mid \mathsf{array}(X, dim)$$

mentre le procedure abbiamo

$$dp \quad ::= \quad \mathsf{nullproc} \mid dp_1; dp_2 \mid \mathsf{proc}\ X(pf)\{c\}$$

nullvar e nullproc sono le dichiarazioni vuote.

La dichiarazione di una variabile associa all'identificatore una *locazione*, e nella dichiarazione di array X è il nome dell'array mentre *dim* è la dimensione, nell'ambiente viene associata al nome X la coppia (locazione del primo elemento, dimensione dell'array). Nella dichiarazione di una procedura $\mathsf{proc}(X, prm, c)$ X è l'identificatore (il nome della procedura), e nell'ambiente si associano al nome il corpo della procedura e il parametro formale (pf),

Espressioni: Le espressioni seguono la seguente grammatica

$$e \quad ::= \quad n \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 \times e_2 \mid \text{true} \mid \text{false} \mid e_1 \wedge e_2 \mid e_1 \vee e_2 \mid \neg e_1 \mid e_1 = e_2 \mid e_1 \leq e_2 \mid X \mid X[e]$$

n indica il numero n, true e false sono i valori base, X è una variabile mentre $X[e]$ è la variabile associata all'i-esima posizione dell'array, dove i è la valutazione dell'espressione e che deve restituire un intero positivo, entrambe restituiscono il valore associato in memoria alla locazione associata al nome, con la differenza che nel caso dell'array viene fatto anche un controllo sulla dimensione.

Comandi: I comandi sono i seguenti

$$c \quad ::= \quad \mid \mathsf{skip} \mid \mathsf{break} \mid X := e \mid X[e_1] := e_2 \mid c_1; c_2 \mid \mathsf{repeat}\ c\ \mathsf{forever} \mid \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2$$

$$\{dv; c\} \mid X(pa)$$

il comando break interrompe la computazione. Viene usato sostanzialmente per interrompere la computazione di un repeat c forever che *ripete* l'esecuzione del comando indefinitamente. Gli altri costrutti sono standard. $X(par)$ indica la chiamata alla procedura X con parametro attuale pa. I blocchi sono delimitati da parentesi graffe e possono avere dichiarazioni locali ma solo di variabili, non di procedure.

Parametri Il parametro (sia formale che attuale) può essere passato per valore o per riferimento. Quindi abbiamo che nei parametri formali (pf) dobbiamo specificare come è passato un parametro, mentre i parametri attuali (pa) possono essere espressioni qualsiasi.

$$pf \quad ::= \quad \mathsf{val}\ X \mid \mathsf{ref}\ X$$

$$pa \quad ::= \quad e$$

Programma: Un programma è fatto da una dichiarazione ed un comando, quindi la sintassi è la seguente:

$$p \quad ::= \quad dv; dp; c$$

La semantica del linguaggio Repeat

Di seguito, per ogni categoria sintattica, vengono date le regole della semantica. Si assume che lo scope sia dinamico. Con ρ indichiamo l'*ambiente* (una funzione tra identificatori ed oggetti denotabili), con σ indichiamo la *memoria* (una funzione da locazioni a valori memorizzabili) ed abbiamo un *flag* con due valori (br o ok) che indica se è stato eseguito un break. $\rho_\perp$ è la funzione ambiente ovunque indefinita e così la $\sigma_\perp$ è la memoria ovunque indefinita. ℓ è una locazione e indica la prima locazione libera. Assumiamo che le locazioni siano interi e che partano da 0.

Programma

$$\frac{\langle dv, (\rho_\perp, 0)\rangle \mapsto_{dv} (\rho', \ell') \qquad \langle dp, (\rho', \ell')\rangle \mapsto_{dp} (\rho, \ell') \qquad \langle c, (\rho, \sigma, \mathsf{ok}, \ell)\rangle \mapsto_c \sigma'}{\langle dv; dp; c, (\rho_\perp, \sigma_\perp)\rangle \mapsto_p (\rho, \sigma')}$$

Dichiarazioni

Per le dichiarazioni di variabili e array abbiamo:

$$\frac{}{\langle \mathsf{nullvar}, (\rho, \ell)\rangle \mapsto_{dv} (\rho, \ell)} \qquad \frac{\langle dv_1, (\rho, \ell)\rangle \mapsto_{dv} (\rho'', \ell'') \qquad \langle dv_2, (\rho'', \ell'')\rangle \mapsto_{dv} (\rho', \ell')}{\langle dv_1; dv_2, (\rho, \ell)\rangle \mapsto_{dv} (\rho', \ell')}$$

$$\frac{}{\langle \mathsf{var}(X), (\rho, \ell)\rangle \mapsto_{dv} (\rho\{\ell/X\}, \ell + 1)} \qquad \frac{}{\langle \mathsf{array}(X, dim), (\rho, \ell)\rangle \mapsto_{dv} (\rho\{(\ell, dim)/X\}, \ell + dim)}$$

mentre le regole per le procedure sono

$$\frac{}{\langle \mathsf{nullproc}, (\rho, \ell)\rangle \mapsto_{dp} (\rho, \ell)} \qquad \frac{\langle dp_1, (\rho, \ell)\rangle \mapsto_{dp} (\rho'', \ell) \qquad \langle dp_2, (\rho'', \ell)\rangle \mapsto_{dp} (\rho', \ell)}{\langle dp_1; dp_2, (\rho, \ell)\rangle \mapsto_{dp} (\rho', \ell)}$$

$$\frac{}{\langle \mathsf{proc}\ X(pf)\{c\}, (\rho, \ell)\rangle \mapsto_{dp} (\rho\{(pf, c)/X\}, \ell)}$$

Espressioni

$$\frac{}{\langle X, (\rho, \sigma)\rangle \mapsto_v \sigma(\rho(X))} \qquad \frac{}{\langle n, (\rho, \sigma)\rangle \mapsto_v \mathbf{n}} \qquad \frac{}{\langle \mathit{true}, (\rho, \sigma)\rangle \mapsto_v \mathbf{true}}$$

$$\frac{}{\langle \mathit{false}, (\rho, \sigma)\rangle \mapsto_v \mathbf{false}} \qquad \frac{\langle e, (\rho, \sigma)\rangle \mapsto_v \mathbf{v}}{\langle \neg e, (\rho, \sigma)\rangle \mapsto_v \neg \mathbf{v}}$$

$$\frac{\rho(X) = (\ell, n) \qquad \langle e, (\rho, \sigma)\rangle \mapsto_v \mathbf{v} \qquad \ell + \mathbf{v} < n}{\langle X[e], (\rho, \sigma)\rangle \mapsto_v \sigma(\ell + \mathbf{v})}$$

$$\frac{\langle e_1, (\rho, \sigma)\rangle \mapsto_v \mathbf{v}_1 \qquad \langle e_2, (\rho, \sigma)\rangle \mapsto_v \mathbf{v}_2}{\langle e_1\ \mathit{op}\ e_2, (\rho, \sigma)\rangle \mapsto_v \mathbf{v}_1\ \mathbf{op}\ \mathbf{v}_2} \quad \mathbf{op}, \mathit{op} \in \{+, -, \times, \leq, =, \wedge, \vee\}$$

Comandi

Nelle regole seguenti $\gamma \in \{\mathbf{ok}, \mathbf{br}\}$. Dato che i comandi ora hanno una sorta di tipo di terminazione (**ok** o **br**) ci sono due regole big step per il ';'.

$$\frac{}{\langle \mathsf{skip}, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma, \mathbf{ok}, \ell)} \qquad \frac{}{\langle \mathsf{break}, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma, \mathbf{br}, \ell)}$$

$$\frac{\langle e, (\rho, \sigma)\rangle \mapsto_v \mathbf{v}}{\langle X := e, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma\{\mathbf{v}/\rho(X)\}, \mathbf{ok}, \ell)}$$

$$\frac{\langle e_0, (\rho, \sigma)\rangle \mapsto_v \mathbf{v}_0 \qquad \langle e, (\rho, \sigma)\rangle \mapsto_v \mathbf{v} \qquad \rho(X) = (\ell, n) \qquad \ell + \mathbf{v} < n}{\langle X[e] := e_0, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma\{\mathbf{v}_0/\ell + \mathbf{v}\}, \mathbf{ok}, \ell)}$$

$$\frac{\langle c_1, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \mathbf{br}, \ell')\rangle}{\langle c_1; c_2, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \mathbf{br}, \ell')}$$

$$\frac{\langle c_1, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma'', \mathbf{ok}, \ell'') \qquad \langle c_2, (\rho, \sigma'', \mathbf{ok}, \ell'')\rangle \mapsto_c (\sigma', \gamma, \ell')}{\langle c_1; c_2, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \gamma)}$$

$$\frac{\langle e, (\rho, \sigma)\rangle \mapsto_v \mathbf{true} \quad \langle c_1, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \gamma, \ell')}{\langle \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \gamma, \ell')} \qquad \frac{\langle e, (\rho, \sigma)\rangle \mapsto_v \mathbf{false} \quad \langle c_2, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \gamma, \ell')}{\langle \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \gamma, \ell')}$$

$$\frac{\langle c, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma'', \mathbf{ok}, \ell'') \qquad \langle \mathsf{repeat}\ c\ \mathsf{forever}, (\rho, \sigma'', \mathbf{ok}, \ell'')\rangle \mapsto_c (\sigma', \mathbf{ok}, \ell')}{\langle \mathsf{repeat}\ c\ \mathsf{forever}, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \mathbf{ok}, \ell')}$$

$$\frac{\langle c, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \mathbf{br}, \ell')}{\langle \mathsf{repeat}\ c\ \mathsf{forever}, (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \mathbf{ok}, \ell')}$$

$$\frac{\langle dv, (\rho, \ell)\rangle \mapsto_{dv} (\rho', \ell') \qquad \langle c, (\rho', \sigma, \mathbf{ok}, \ell')\rangle \mapsto_c (\sigma', \gamma, \ell'')}{\langle \mathsf{block}(dv, c), (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \gamma, \ell'')}$$

$$\frac{\rho(X) = (\mathsf{val}\ Y, c) \qquad \langle e, (\rho, \sigma)\rangle \mapsto_v \mathbf{v} \qquad \langle c, (\rho'\{\ell/Y\}, \sigma\{\mathbf{v}/\ell\}, \mathbf{ok}, \ell + 1)\rangle \mapsto_c (\sigma', \gamma, \ell')}{\langle X(e), (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \gamma, \ell')}$$

$$\frac{\rho(X) = (\mathsf{ref}\ Y, c) \qquad \langle c, (\rho'\{\rho(Z)/Y\}, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \gamma, \ell')}{\langle X(Z), (\rho, \sigma, \mathbf{ok}, \ell)\rangle \mapsto_c (\sigma', \gamma, \ell')}$$