Progetto del Corso di

Laboratorio di Linguaggi

Anno Accademico 2018/2019

Estensione della Semantica Statica e Dinamica di un Semplice Linguaggio Imperativo

Assegnamento

Si estenda l'interprete StaticImp presentato a lezione e scaricabile dall'elearning seguendo il percorso `/Project/Base/staticimp.zip` con i seguenti nuovi comandi:

- **Ciclo `for`.** Il ciclo `for` è una struttura di controllo iterativa la cui sintassi è data dalla seguente regola:

$$\texttt{for (}\mathit{assign}\texttt{; }\mathit{exp}\texttt{; }\mathit{com}_0\texttt{) \{ }\mathit{com}_1\texttt{ \}}$$

dove *assign* è un assegnamento arbitrario. La semantica dinamica di un generico ciclo `for` è equivalente al seguente comando:

$$\mathit{assign}\texttt{; while (}\mathit{exp}\texttt{) \{ }\mathit{com}_1\texttt{; }\mathit{com}_0\texttt{ \}}$$

Affinchè il costrutto `for` sia ben tipato, è necessario che *assign*, com_0, com_1, ed *exp* siano ben tipati e che *exp* sia di tipo booleano.

- **Ciclo `do-while`.** Si implementi il costrutto iterativo `do-while` che esegue il suo corpo almeno una volta. La sua sintassi è

$$\texttt{do \{ }\mathit{com}\texttt{ \} while (}\mathit{exp}\texttt{)}$$

e la sua semantica risulta equivalente al seguente comando:

$$\mathit{com}\texttt{; while (}\mathit{exp}\texttt{) \{ }\mathit{com}\texttt{ \}}$$

Il costrutto `do-while` è ben tipato se *com* ed *exp* risultano ben tipati e se quest'ultima è di tipo booleano.

- **Operatore condizionale `if-then-elseif-else`.** Si modifichi/estenda l'operatore condizionale `if-then-else` già presente nel linguaggio secondo la seguente regola sintattica:

$$\begin{aligned}&\texttt{if (}\mathit{exp}_0\texttt{) then \{ }\mathit{com}_0\texttt{ \}}\\&\texttt{elseif (}\mathit{exp}_1\texttt{) \{ }\mathit{com}_1\texttt{ \}}\\&\ldots\\&\texttt{elseif (}\mathit{exp}_n\texttt{) \{ }\mathit{com}_n\texttt{ \}}\\&\texttt{else \{ }\mathit{com}_{n+1}\texttt{ \}}\end{aligned}$$

I rami `elseif` ed `else` sono opzionali e possono essere omessi, mentre il ramo `if` è obbligatorio affinchè il costrutto risulti ben formato. La semantica di un generico `if-then-elseif-else` è equivalente al seguente

comando qualora sia presente almeno un ramo `elseif` o `else`:

```
if (exp_0) then { com_0 }
else {
    if (exp_1) then { com_1 }
    else {
        if (exp_2) then { com_2 }
        else {
            ...
        }
    }
}
```

Nel caso in cui sia presente solamente il ramo `if`, l'equivalenza è data rispetto al comando

```
if (exp_0) then { com_0 } else { skip }
```

L'operatore condizionale risulta ben tipato quando tutte le espressioni exp_i ed i comandi com_i sono ben tipati. In particolare, le espressioni devono avere tipo booleano.

▪ **Scelta non deterministica `nd`.** L'operatore per la scelta non deterministica tra due comandi ha la seguente sintassi:

```
nd(com_0, com_1)
```

La sua semantica intuitiva è di eseguire *casualmente* uno tra i due comandi com_0 e com_1. Affinchè risulti ben tipato, devono essere ben tipati i due comandi utilizzati come argomenti.

Linee Guida

Dopo aver scaricato ed importato il progetto di partenza `staticimp.zip` in IntelliJ, lo svolgimento dell'elaborato può essere suddiviso in tre fasi:

Estensione della grammatica: per ognuno dei costrutti da implementare, si estenda la grammatica del linguaggio in modo consistente con le regole sintattiche mostrate nella sezione precedente.

Implementazione della semantica statica: si esegua l'override dei nuovi metodi generati da ANTLR in seguito alla modifica della grammatica nella classe `StaticIntImp` al fine di implementarne la semantica statica.

Implementazione della semantica dinamica: si proceda come nel punto precedente al fine di implementare la semantica dinamica dei nuovi costrutti eseguendo l'override dei metodi nella classe `DynamicIntImp`.

Testing

Una volta terminata l'implementazione, si eseguano dei test sia rispetto a programmi ben tipati che non per verificarne la correttezza. In particolare, una serie di programmi sono già stati caricati sull'elearning al percorso `/Project/Test`. I programmi all'interno della sottocartella `Well-Typed` sono programmi ben tipati che devono essere eseguiti dall'interprete senza nessun errore. Invece, i programmi all'interno della sottocartella `Bad-Typed` sono programmi mal tipati che devono stampare a video un errore durante la fase dell'analisi dei tipi (semantica statica).

Per semplificarvi la fase finale di testing, la classe `Test` nel package `org.univr.staticimp` al percorso `src/test` esegue tutti i test in automatico. Per lanciarli, è sufficiente cliccare col destro sul file della classe `Test` (all'interno di IntelliJ) e scegliere `Run 'Test'`. Se tutti i test terminano correttamente, il progetto viene considerato corretto.[1] In caso contrario, la consegna di un progetto non funzionante viene valutata 0 punti a prescindere dalle modalità di valutazione spiegate nella prossima sezione.

[1] Affinchè tutto funzioni come previsto, è importante **non** modificare la classe `Test`, il file `pom.xml`, e il codice sorgente dei programmi nella cartella `resources`.

Modalità di Consegna e Valutazione

Una volta terminato il progetto, si proceda alla consegna nel seguente modo: si esporti il progetto da IntelliJ (`File > Export to Zip file...`) e lo si invii a `samuele.buro@univr.it` indicando il proprio nome, cognome, e matricola. Se il progetto è stato svolto in coppia, inviate una singola email contenente anche i dati del vostro collega che metterete in cc.

Il progetto deve essere consegnato almeno una settimana prima del compito scritto che vorrete sostenere ed annulla un eventuale altro scritto svolto in precedenza. Se il progetto funziona correttamente (ovvero se esegue come previsto su tutti i test caricati sull'elearning), avrete la possibilità di rispondere ad una domanda riguardante l'implementazione del vostro progetto il giorno che vi presenterete a fare lo scritto. I punti che potrete ottenere sono

- 4, se il progetto è stato consegnato entro il 28/01 (indipendentemente da quando poi andrete a sostenere l'esame); oppure
- 3, negli altri casi.

Il progetto è consegnabile una sola volta (prima dello scritto, a meno che non decidiate di farvi annullare il voto) ed avrete una sola possibilità di rispondere alla domanda sul progetto il giorno che vi presentere a fare lo scritto. Se la risposta non viene data o se prendete un punteggio che non vi soddisfa, non avrete la possibilità di ripresentarvi agli appelli successivi per rispondere nuovamente alla domanda sul progetto. Ovviamente, potrete ripetere lo scritto senza perdere il punteggio ottenuto sulla domanda riguardante il progetto.

I punti ottenuti dal progetto (max. 4) si sommano a quelli dello scritto (max. 27). Se la somma supera il 28 e lo studente vuole registrare un voto strettamente maggiore di 28, dovrà sostenere un esame orale, altrimenti il voto verrà troncato a 28. Una regola simile vale anche per chi non svolge il progetto: i punti ottenibili per chi ha deciso di non svolgere il progetto sono i soli dati dallo scritto (max. 27), e chi ha un voto superiore a 26 e vuole registrare un voto strettamente maggiore di 26, dovrà sostenere un esame orale.

Inoltre, per poter registrare il voto è necessario aver preso la sufficienza nel compito scritto.

Per venire incontro agli studenti che si sono iscritti per la prima volta al terzo anno nel 2017/2018 oppure precedenti (ovvero quelli che potenzialmente non stanno seguendo il corso per la prima volta), vi è la possibilità di scegliere se svolgere il progetto oppure se rispondere ad una domanda teorica sulla semantica operazionale indipendente dal progetto. Per tutti gli altri (ovvero chi si è iscritto al terzo anno nel 2018/2019) il progetto è l'unico modo per ottenere i suddetti punti.

Note su Come Importare il Progetto in IntelliJ

1. Scaricate il file `staticimp.zip` da `/Project/Base/staticimp.zip` (non scaricate il file con lo stesso nome contenuto nel materiale della quinta lezione, in quanto non contiene le classi e la configurazione per i test).
2. Estraete i file contenuti e prestate attenzione che non vengano create due cartelle annidate entrambe con il nome `staticimp`. Al termine dell'estrazione dovrete avere una cartella `staticimp` contenente una cartella `src` e il file `pom.xml`.
3. Dalla finestra iniziale di IntelliJ, scegliete `Import Project`. Se IntelliJ non vi apre la finestra iniziale ma l'ultimo progetto aperto, andate su `File > Close Project` e tornerete alla finestra iniziale.
4. Selezionate la cartella estratta `staticimp` e confermate.
5. Selezionate `Import project from external model`, selezionate `Maven` e proseguite.
6. Spuntate `Import Maven projects automatically` e proseguite due volte.
7. Selezionate l'SDK di Java 8, proseguite, e terminate.