Linee guida per il progetto e per l'esame di *Linguaggi e compilatori*, parte I: front-end

Progetto

Il progetto può partire (anche se non obbligatoriamente) dal prototipo base fornito dal docente (cartella Kaleidoscope1.0 compressa su GDrive), che implementa la primissima versione del linguaggio, fino al costrutto *ifexpr* escluso.

Le seguenti estensioni potranno portare alla valutazione massima indicata per ciascuna di esse. Si intende che ogni estensione presuppone tutte quelle precedenti.

Estensione 1 (punteggio max **18**). Il front-ent deve supportare il costrutto *ifexpr* e almeno un operatore relazionale da poter usare nelle espressioni che controllano il condizionale (<, <=, ecc.). Per chiarezza, riportiamo la sintassi del costrutti *ifexpr* (qui e nel seguito i simboli non terminali sono indicati in corsivo)

expr → ... | *ifexpr*
ifexpr → if *expr* then *expr* else *expr* end

Estensione 2 (punteggio max **21**). Il front-ent deve supportare tutti gli operatori relazionali, l'operatore "-" unario (esempio: -3+2, che deve avere come risultato -1 e non -5) e le espressioni composte, ovvero espressioni formate da sequenze di espressioni separate dall'operatore ":". Per aggiungerle alla sintassi è sufficiente estendere le produzioni per le espressioni nel modo seguente

expr → ... | *expr* : *expr*

attribuendo all'operatore ":" (trattato come se fosse un comune operatore binario) l apiù bassa priorità. Il valore di una espressione composta si definisce semplicemente come il valore dell'ultima espressione nella sequenza.

Estensione 3 (punteggio max **24**). Il front-ent deve supportare il costrutto *forexpr*

expr → ... | for*expr*
forexpr → for id = *expr* , *expr step* in *expr* end
step → eps | , *expr*

Il valore di una espressione for è il valore dell'espressione che definisce il suo body. Se lo step non è presente è come se fosse 1.

Estensione 4 (punteggio max **27**). Il front-ent deve supportare variabili "mutable", assegnamento e blocchi con dichiarazioni (*varexpr*). Queste estensioni richiedono un certo impegno per la piena comprensione. La sintassi del linguaggio si può modificare nel modo seguente. Si noti che gli assegnamenti in Kaleidoscope sono

ammessi solo come modo per alterare il valore di una variabile già definita nello scope che include l'assegnamento. Come sappiamo, questo vincolo non è esprimibile in una grammatica context-free. Assegnamenti possono trovarsi solo all'interno di funzioni o di varexpression ma questo viene determinato non dal parser dal generatore di codice intermedio (che, in questo caso, effettua anche un controllo semantico). Gli assegnamenti in Kaleidoscope sono definiti come espressioni il cui valore è il valore della parte destra.

expr → ... | *varexpr* | *assignment*
varexpr → var *varlist* in *expr* end
varlist → *pair* | *pair* , *varlist*
pair → id | id = *expr*
assignment → *id* = *expr*

Il valore di una *varexpr* è definito come il valore della espressione che rappresenta il suo body (quella dopo la keyword in)

Estensione 5 (punteggio max **30**) Il front-end deve supportare il costrutto *whileexp,* di cui lo studente è chiamato a formulare adeguata sintassi

Estensione 6 (punteggio max **30L**) Il front-end deve supportare la struttura dati "array di double" (statica) e, chiaramente, l'accesso ai singoli elementi. Anche di questa estensione lo studente è chiamato a formulare adeguata sintassi.

Nota 1. Tutti le estensioni al front-end devono produrre in output una valida rappresentazione intermedia LLVM, tale cioè che possa essere soggetta a compilazione in codice oggetto. Al riguardo, il prototipo contiene già le istruzioni per produrre tale codice oggetto. Si consiglia di utilizzare almeno tale codice per la verifica di correttezza della IR prodotta.

Nota 2. La valutazione massima richiede che lo studente sia in grado di dare conto con precisione del codice prodotto. Questo è particolarmente evidente nel caso delle prime estensioni che, di fatto, sono già state viste a lezione e/o sono ben documentate nel tutorial.

Nota 3. Nel tutorial vengono estesamente utilizzati puntatori "univoci", ovvero oggetti del tipo `STD::UNIQUE_POINTER<<TYPE>>`, dove `<TYPE>` è di volta in volta la classe `EXPRAST`, `NUMBEREXPRAST`, …, `LLVMCONTEXT`, `IRBUILDER<>`. L'utilizzo di unique_pointer comporta sensibili vantaggi nel senso che riduce i rischi legati alla gestione esplicita dei puntatori (memory leak, riferimenti pendenti). Tuttavia, per non accrescere ulteriormente la quantità di materiale innovativo, nel prototipo "base" Kaleidoscope1.0 il loro utilizzo è stato volutamente evitato. Lo studente è libero di re-introdurli o di continuare a lavorare con puntatori "classici".

Esame

L’esame della prima parte richiede lo svolgimento del progetto, la sua discussione e una prova orale. La discussione del progetto e la prova orale possono, a richiesta, essere svolte in tempi separati. Questa non è però l’opzione consigliata dal docente ed è ammessa senza che sia necessario esporre motivazioni solo nel caso di progetto congiunto, in cui uno solo dei sue “candidati” vuole anche svolgere l’orale. Negli altri casi la richiesta va motivata ed è a discrezione del docente ammetterla. In ogni caso, se il progetto è congiunto, i candidati devono discuterlo nello stesso giorno.
Il progetto può essere svolto singolarmente o in gruppi di due persone.
Le date pubblicate su esse3 si riferiscono a sessioni d’esame orale in cui è possibile discutere il progetto e/o svolgere la prova orale, secondo quanto illustrato sopra.
Per la valutazione, si tenga presente che la parte di front-end vale 50% del totale e, similmente, progetto e orale hanno lo stesso peso (quindi 25% rispetto al voto che verrà verbalizzato).