Homework 9
CS420/KAIST Spring 1999
이 광근
Due: May 27 (목요일), 24:00

1999년 5월 21일

Exercise 1 "Run tcoML programs on SECK machine"

수업시간에 정의한 tcoML-to-SECK 를 구현해서 지금까지 구축된 tcoML 컴파일러에 장착한다. SECK 코드 생성과정은 컴파일러에서 타입검증을 거친 후에 있게되므로, 만들어진 SECK코드는 실행중에 잘못되지 말아야 한다.

SECK코드 실행기(interpreter)는 TA가 곧 공개할 것이므로 여러분들의 SECK 코드 생성결과를 실행해 볼 수 있을것이다. □

Exercise 2 "Fix the millenium bug!" **30pt** (프로그램 분석의 맛배기)

6개월후에 발생할 "2000년 문제"를 해결하려면 프로그램에서 년도를 표현하는 두자리 변수들과 관계되는 모든 변수를 찾아내어 네자리 변수로 확장해 주어야한다.

COBOL이라는 언어에서는 숫자를 보관할 변수를 선언할때 자릿수를 지정하게 되어있는데, 예를들어

```
year 99
```

라고 선언해 놓으면, `year`라는 변수는 두자리 정수만 가질 수 있는 변수가 된다. 두자리 년도 데이타를 화일에서 `year`로 읽어들여서 두자리만을 가지고 연산을 진행하게 되는 것이다.

이 문제를 해결하는 한 방법은 `year`를 네자리로 선언하고

```
year 9999
```

화일에서 읽어들인 두자리 년도값 `year`에 1900이나 2000을 적절히 더하는 코드를 넣어주는 것이다:

```
assign year year+1900 혹은
assign year year+2000
```

그러나, 이것으로는 충분하지 않다. 변수 `year`가 네자리로 늘었기 때문에 예전의 두자리 `year`와 "연쇄 반응"하는 모든 변수들도 찾아가서 네자리로 늘려잡아놔야 하기때문이다.

자, 다음의 tcoCOBOL 을 생각하자:

$$
\begin{array}{lcl}
\mathit{pgm} & ::= & \mathit{decl}+\ \ \mathit{cmd}+ \\
\mathit{decl} & ::= & x\texttt{:}\ \mathit{number} \\
\mathit{cmd} & ::= & \mathit{label}\ \ \mathit{stmt} \\
 & | & \mathit{stmt} \\
\mathit{stmt} & ::= & \texttt{input}\ x \\
 & | & \texttt{output}\ x \\
 & | & \texttt{assign}\ x\ \ \mathit{expr} \\
 & | & \texttt{if}\ \mathit{expr}\ \ \mathit{stmt}\ \ \mathit{stmt} \\
 & | & \texttt{goto}\ \mathit{label} \\
\mathit{expr} & ::= & x \\
 & | & \mathit{number} \\
 & | & \mathit{expr}\ \texttt{+}\ \mathit{expr} \\
 & | & \mathit{expr}\ \texttt{=}\ \mathit{expr}
\end{array}
$$

이번 숙제는, 위의 tcoCOBOL로 짜여진 프로그램 pgm에서 지정된 변수 year에 "연쇄반응"하는 모든 변수 millenium(pgm, year)를 찾는 프로그램 millenium을 작성하는 것이다:

```
millenium:  Cobol.ast * Cobol.id -> Cobol.id set
```

변수 year에 연쇄반응하는 변수들은

- {year} 이거나
- 지금까지 연쇄반응하는 변수들을 이용해서 값이 결정되는 변수들이다.

즉, 다음의 방정식을 만족하는 집합 S 이다:

$$S = \{\texttt{year}\} \cup \mathit{Assignee}(S)$$

where

$$\mathit{Assignee} : 2^{Vars} \to 2^{Vars}$$
$$\mathit{Assignee}(X) = \{y | \text{“}\texttt{assign}\ y\ \ \mathit{expr}\text{''} \text{ in program} \land\ \mathit{expr} \text{ has } x \in X\}.$$

Structure Cobol의 시그내쳐는:

```
signature COBOL =
 sig
  type id
  type label
  datatype ast = PGM of (id * int) list * cmd list
       and cmd = L of label * stmt | S of stmt
       and stmt = INPUT of id
                | OUTPUT of id
                | ASSIGN of id * expr
                | IF of expr * stmt * stmt
                | GOTO of label
       and expr = VAR of id
                | NUM of int
                | PLUS of expr * expr
                | EQUAL of expr * expr
 end
```

□