

Homework 4

SNU 4190.310, 2012 가을

이 광근

Program Due: 10/26(금) 24:00

Essay Due: 10/29(월), 09:30

이번 숙제의 목적은:

- 상식적인 수준에서 디자인된 명령형 언어의 대표적인 C언어의 역사와, 컴퓨터 실행(기계적인 계산)과 상위 논리의 관계, 프론티어들의 프로그래밍 언어 사용경향에 대한 자료등을 읽고 느낀 바를 글로 쓰기.
- 언어사이의 자동 번역기를 제작해보기.
- 메모리 재활용의 기본개념을 구현해보기.
- 앞으로 프로그래밍 언어 구현에서 넘어야 할 산이, 상식만으로는 넘기 어렵다는 것을 겪어보기.

Exercise 1 (30pts) “논술 에세이”

강의 홈페이지의 읽을거리 [Part I]에 매달린 글들을 읽고(비디오 포함) 리포트로 작성해서 제출합니다.

- 논술의 구성은: 읽은 내용 정리 30%, 읽고 느낀점 70%로.
- 각 단락은 두괄식으로. 두괄식이란, 단락의 결론을 단락의 첫 문장으로 가져오는 것을 말합니다. 단락 내용을 정리한 문장(topic sentence)이 단락의 첫 문장. 작성한 에세이 단락들의 첫 문장들만을 읽어도 논술의 흐름이 부드럽게 되는 지 확인.

- A4용지 총 10 페이지를 넘기지 말것.
- 반드시 컴퓨터로 출력해서 제출.
- 10/29(월) 수업시간에 제출. No delay acceptable.

□

Exercise 2 (40pts) “SM5”

K--(교재 4.3) 프로그램들을 가상 기계(abstract machine)인 SM5에서 실행될 수 있도록 번역하는 번역기를 제작한다.

SM5는 가상의 기계이다. “SM”은 “Stack Machine”을 뜻하고, “5”는 그 기계의 부품이 5개이기 때문이다:

$$(S, M, E, C, K)$$

S 는 스택, M 은 메모리, E 는 환경, C 는 명령어, K 는 남은 할 일(“continuation”이라고 부름)을 뜻하고 다음 집합들의 원소이다:

$$\begin{aligned}
 S &\in \text{Stack} = \text{Svalue list} \\
 M &\in \text{Memory} = \text{Loc} \rightarrow \text{Value} \\
 E &\in \text{Environment} = (\text{Var} \times (\text{Loc} + \text{Proc})) \text{ list} \\
 C &\in \text{Command} = \text{Cmd list} \\
 K &\in \text{Continuation} = (\text{Command} \times \text{Environment}) \text{ list} \\
 \\
 v &\in \text{Value} = \text{Integer} + \text{Bool} + \{\cdot\} + \text{Record} + \text{Loc} \\
 x &\in \text{Var} \\
 \langle a, o \rangle, l &\in \text{Loc} = \text{Base} \times \text{Offset} \\
 &\text{Offset} = \text{Integer} \\
 z &\in \text{Integer} \\
 b &\in \text{Bool} \\
 r &\in \text{Record} = (\text{Var} \times \text{Loc}) \text{ list} \\
 w &\in \text{Svalue} = \text{Value} + \text{Proc} + (\text{Var} \times \text{Loc}) \quad (* \text{ stackable values } *) \\
 p &\in \text{Proc} = \text{Var} \times \text{Command} \times \text{Environment} \\
 \text{Cmd} &= \{\text{push } v, \text{push } x, \text{push}(x, C), \\
 &\quad \text{pop}, \text{store}, \text{load}, \text{jtr}(C, C), \\
 &\quad \text{malloc}, \text{box } z, \text{unbox } x, \text{bind } x, \text{unbind}, \text{get}, \text{put}, \text{call}, \\
 &\quad \text{add}, \text{sub}, \text{mul}, \text{div}, \text{eq}, \text{less}, \text{not}\}
 \end{aligned}$$

기계의 작동은 다음과 같이 기계의 상태가 변화하는 과정으로 정의할 수 있다:

$$(S, M, E, C, K) \Rightarrow (S', M', E', C', K')$$

언제 어떻게 위의 기계작동의 한 스텝(\Rightarrow)이 일어나는 지는 다음과 같다:

$$\begin{array}{l} (S, \quad \quad \quad M, \quad E, \quad \quad \text{push } v :: C, \quad K) \\ \Rightarrow (v :: S, \quad \quad \quad M, \quad E, \quad \quad \quad \quad \quad \quad C, \quad K) \end{array}$$

$$\begin{array}{l} (S, \quad \quad \quad M, \quad E, \quad \quad \text{push } x :: C, \quad K) \\ \Rightarrow (w :: S, \quad \quad \quad M, \quad E, \quad \quad \quad \quad \quad \quad C, \quad K) \text{ if } (x, w) \text{ is the first such entry in } E \end{array}$$

$$\begin{array}{l} (S, \quad \quad \quad M, \quad E, \quad \text{push } (x, C') :: C, \quad K) \\ \Rightarrow ((x, C', E) :: S, \quad \quad \quad M, \quad E, \quad \quad \quad \quad \quad \quad C, \quad K) \end{array}$$

$$\begin{array}{l} (w :: S, \quad \quad \quad M, \quad E, \quad \quad \quad \text{pop} :: C, \quad K) \\ \Rightarrow (S, \quad \quad \quad \quad \quad \quad M, \quad E, \quad \quad \quad \quad \quad \quad C, \quad K) \end{array}$$

$$\begin{array}{l} (l :: v :: S, \quad \quad \quad M, \quad E, \quad \quad \quad \text{store} :: C, \quad K) \\ \Rightarrow (S, \quad \quad \quad M\{l \mapsto v\}, \quad E, \quad \quad \quad \quad \quad \quad C, \quad K) \end{array}$$

$$\begin{array}{l} (l :: S, \quad \quad \quad M, \quad E, \quad \quad \quad \text{load} :: C, \quad K) \\ \Rightarrow (M(l) :: S, \quad \quad \quad M, \quad E, \quad \quad \quad \quad \quad \quad C, \quad K) \end{array}$$

\Rightarrow	$(true :: S,$	$M,$	$E,$	$\mathbf{jtr}(C_1, C_2) :: C, K)$	
	$(S,$	$M,$	$E,$	$C_1 :: C, K)$	
\Rightarrow	$(false :: S,$	$M,$	$E,$	$\mathbf{jtr}(C_1, C_2) :: C, K)$	
	$(S,$	$M,$	$E,$	$C_2 :: C, K)$	
\Rightarrow	$(S,$	$M,$	$E,$	$\mathbf{malloc} :: C, K)$	
	$(\langle a, 0 \rangle :: S,$	$M,$	$E,$	$C, K)$	$\text{new } a$
\Rightarrow	$(w_1 :: \dots :: w_z :: S,$	$M,$	$E,$	$\mathbf{box } z :: C, K)$	
	$([w_1, \dots, w_z] :: S,$	$M,$	$E,$	$C, K)$	
\Rightarrow	$([w_1, \dots, w_z] :: S,$	$M,$	$E,$	$\mathbf{unbox } x :: C, K)$	
	$(v :: S,$	$M,$	$E,$	$C, K)$	$w_k = (x, v), 1 \leq k \leq z$
\Rightarrow	$(w :: S,$	$M,$	$E,$	$\mathbf{bind } x :: C, K)$	
	$(S,$	$M,$	$(x, w) :: E,$	$C, K)$	
\Rightarrow	$(S,$	$M,$	$(x, w) :: E,$	$\mathbf{unbind} :: C, K)$	
	$((x, w) :: S,$	$M,$	$E,$	$C, K)$	
\Rightarrow	$(l :: v :: (x, C', E') :: S,$	$M,$	$E,$	$\mathbf{call} :: C, K)$	
	$(S,$	$M\{l \mapsto v\},$	$(x, l) :: E',$	$C', (C, E) :: K)$	
\Rightarrow	$(S,$	$M,$	$E,$	$\mathbf{empty}, (C, E') :: K)$	
	$(S,$	$M,$	$E',$	$C, K)$	
\Rightarrow	$(S,$	$M,$	$E,$	$\mathbf{get} :: C, K)$	
	$(z :: S,$	$M,$	$E,$	$C, K)$	$\text{read } z \text{ from outside}$
\Rightarrow	$(z :: S,$	$M,$	$E,$	$\mathbf{put} :: C, K)$	
	$(S,$	$M,$	$E,$	$C, K)$	$\text{print } z \text{ and newline}$

$$\begin{aligned}
& (v_2 :: v_1 :: S, \quad M, \quad E, \quad \mathbf{add} :: C, \quad K) \\
\Rightarrow & (\mathbf{plus}(v_1, v_2) :: S, \quad M, \quad E, \quad C, \quad K) \\
& (v_2 :: v_1 :: S, \quad M, \quad E, \quad \mathbf{sub} :: C, \quad K) \\
\Rightarrow & (\mathbf{minus}(v_1, v_2) :: S, \quad M, \quad E, \quad C, \quad K) \\
& (z_2 :: z_1 :: S, \quad M, \quad E, \quad \mathbf{mul} :: C, \quad K) \\
\Rightarrow & ((z_1 * z_2) :: S, \quad M, \quad E, \quad C, \quad K) \quad \text{similar for div} \\
& (v_2 :: v_1 :: S, \quad M, \quad E, \quad \mathbf{eq} :: C, \quad K) \\
\Rightarrow & (\mathbf{equal}(v_1, v_2) :: S, \quad M, \quad E, \quad C, \quad K) \\
& (v_2 :: v_1 :: S, \quad M, \quad E, \quad \mathbf{less} :: C, \quad K) \\
\Rightarrow & (\mathbf{less}(v_1, v_2) :: S, \quad M, \quad E, \quad C, \quad K) \\
& (b :: S, \quad M, \quad E, \quad \mathbf{not} :: C, \quad K) \\
\Rightarrow & (\neg b :: S, \quad M, \quad E, \quad C, \quad K)
\end{aligned}$$

$$\begin{aligned}
\mathbf{less}(z_1, z_2) &= z_1 < z_2 \\
\mathbf{plus}(z_1, z_2) &= z_1 + z_2 \\
\mathbf{plus}(\langle a, z_1 \rangle, z_2) &= \langle a, z_1 + z_2 \rangle \quad \text{if } z_1 + z_2 \geq 0 \\
\mathbf{plus}(z_1, \langle a, z_2 \rangle) &= \langle a, z_1 + z_2 \rangle \quad \text{if } z_1 + z_2 \geq 0 \\
\mathbf{minus}(z_1, z_2) &= z_1 - z_2 \\
\mathbf{minus}(\langle a, z_1 \rangle, z_2) &= \langle a, z_1 - z_2 \rangle \quad \text{if } z_1 - z_2 \geq 0 \\
\mathbf{equal}(z_1, z_2) &= z_1 = z_2 \\
\mathbf{equal}(b_1, b_2) &= b_1 = b_2 \\
\mathbf{equal}(\cdot, \cdot) &= \mathbf{true} \\
\mathbf{equal}(r_1, r_2) &= (\forall \langle x, l \rangle \in r_1 : \langle x, l \rangle \in r_2) \wedge (\forall \langle x, l \rangle \in r_2 : \langle x, l \rangle \in r_1) \\
\mathbf{equal}(\langle a_1, z_1 \rangle, \langle a_2, z_2 \rangle) &= a_1 = a_2 \wedge z_1 = z_2 \\
\mathbf{equal}(-, -) &= \mathbf{false}
\end{aligned}$$

SM5의 프로그램 C 를 실행한다는 것은, C 만 가지고 있는 빈 기계상태를 위에서 정의한 방식으로 변환해 간다는 뜻이다:

$$(\mathbf{empty}, \mathbf{empty}, \mathbf{empty}, C, \mathbf{empty}) \Rightarrow \dots \Rightarrow \dots$$

예를들어,

```
push 1 :: push 2 :: add :: put :: empty
```

는 K-- 프로그램 `write 1+2`과 같은 일을 하게 된다.

여러분이 할 것은, 잘 돌아가는 K-- 프로그램을 입력으로 받아서 같은 일을 하는 SM5 프로그램으로 변환하는 함수

```
trans: K.program -> Sm5.command
```

를 작성하는 것이다.

`trans`가 제대로 정의되었는지는, K-- 프로그램 E 에 대해서, $K.run(E)$ 와 $Sm5.run(trans(E))$ 을 실행해서 확인할 수 있을 것이다.

모듈 `Sm5`, 모듈 `K`, 그리고 K--의 파서는 제공된다(TA 페이지 참고). □

Exercise 3 (40pts) “SM5 Limited = SM5 + 메모리 재활용”

SM5 메모리에서는 무한히 많은 새로운 주소가 샘솟을 수 없다.

이제, SM5의 메모리는 $8K(2^{13})$ 개의 주소만 있다고 하자. 위의 문제에서 주어진 모듈 `Sm5`를 뜯어 고쳐서, `malloc`할 것이 더이상 없을 때 메모리를 재활용하는 함수 `gc`를 장착하라. 즉,

$$(S, M, E, malloc :: C, K) \Rightarrow (l :: S, M, E, C, K) \text{ new } l$$

이 아래와 같이 변경될 것이다:

$$\begin{aligned} (S, M, E, malloc :: C, K) &\Rightarrow (l :: S, M, E, C, K) \quad \text{new } l, \text{ if } |\text{dom}M| < 2^{13} \\ (S, M, E, malloc :: C, K) &\Rightarrow (l :: S, gc(\cdot\cdot\cdot), E, C, K) \quad \text{recycled } l, \text{ if } |\text{dom}M| = 2^{13} \end{aligned}$$

재활용함수 `gc`는 실제 구현보다 훨씬 간단하다. 현재 메모리에서 미래에 사용할 수 있는 부분만을 모으면 될 것이다. 그러한 부분들은 현재 기계 상태의 두 개의 부품(□와 □)에서 부터 도달 가능한 모든 메모리 주소들이 될 것이다. □

Exercise 4 (30pts) “탐사 준비”

탐사해야 할 지역의 지도를 보고 탐사를 성공리에 마치기 위해 필요한 최소의 준비물을 알아내는 프로그램을 작성해 보자.

탐사는 지도에 나타난 길을 따라 이동하면서 길에 놓인 보물상자를 열고 보물을 모아가는 것이고, 모든 보물이 모아지면 그 탐사는 성공한 것이다. 준비물은 모든 보물상자를 열 수 있는 열쇠들이다.

보물상자와 열쇠:

- 보물상자에는 고유의 알파벳 이름이 표시되어 있다.
- 이름없이 “*”라고 적혀있는 보물상자도 있다.
- 같은 이름의 보물 상자는 같은 열쇠로 열린다.
- 하나의 열쇠는 외갈래 혹은 두갈래로 갈라진 가지구조(tree)이다.
- 열쇠는 반복해서 사용할 수 있다.

보물상자와 열쇠를 OCaml 타입으로 정의하면,

```
type treasure = StarBox | NameBox of string
type key = Bar | Node of key * key
```

탐사지도:

- 시작 지점은 하나이다.
- 길들은 모두 외길이거나 두 갈래로 나뉘어 진다.
- 보물상자들은 모두 막다른 골목의 끝에 있다.
- 갔던 길을 되돌아 오지 않고 왔던 곳으로 다시 오는 방법은 없다(tree).
- 길목에 세워진 안내판에는 앞으로 만날 보물상자의 알파벳 이름이 쓰여져 있다.
- 모든 안내판의 이름은 모두 다르다.

탐사지도를 OCaml 타입으로 정의하면,

```
type map = End of treasure
         | Branch of map * map
         | Guide of string * map
```

보물상자마다 필요한 열쇠의 모양은 보물상자의 위치가 전체 탐사지도에서 어디냐에 따라 결정되는데, 지도에서 각 지역이 암시하는 열쇠의 모양은 다음의 조건으로 결정된다:

현재위치(지도) e	위치의 뜻	열쇠모양의 조건
*	* 보물상자	- (Bar)
x	x 라는 이름의 보물상자	현재 위치에서 x 를 열어줄 열쇠 모양
$\boxed{x}e_1$	안내판 \boxed{x} 이 앞에 있는 지도 e_1	e_1 안에서 만날 보물상자 x 의 열쇠가 α 이고 e_1 의 시작점이 암시하는 열쇠모양을 β 라고 하면, 현재 위치가 암시하는 열쇠모양은 (α, β) (왼쪽까지 α , 오른쪽까지 β).
$e_1 e_2$	e_1 과 e_2 로 갈라지는 갈림길	e_1 의 시작점이 암시하는 열쇠모양은 (α, β) 이어야 하고 e_2 의 시작점이 암시하는 열쇠모양은 α 이어야 한다. 이때, 현재 위치가 암시하는 열쇠모양은 β .

예를들어, 각 지도를 성공적으로 탐험할 최소의(열쇠들 크기의 합을 기준으로) 열쇠꾸러미는 다음과 같다:

1. 지도 x 에는 $\{-\}$.
2. 지도 $\boxed{x}x$ 에는 $\{-\}$.
3. 지도 $(\boxed{x}x)|*$ 에는 $\{-\}$.
4. 지도 $(\boxed{x}(x|x))|*$ 를 성공적으로 탐험하는 것은 불가능.
5. 지도 $(\boxed{x}x)|((\boxed{y}y)|*)$ 에는 $\{-\}$.
6. 지도 $(\boxed{x}x)|(\boxed{y}y)$ 에는 $\{-, (-, -)\}$.
7. 지도 $x|*$ 에는 $\{-, (-, -)\}$.

다음의 타입에 맞도록, 위와같은 일을 하는 `getReady` 함수

`getReady: map → key list`

를 정의하기 바랍니다.

□