Home Project 2
SNU 010.142 Basics in Computing, Spring 2006

due: 5/23 24:00

이번 프로젝트에서는 다양한 나무구조가 필요한 경우를 만나게 됩니다. 모든 경우마다 짝(pair)이라는 개념을 가지고 표현하는 방안을 고안하고 그것을 C로 구현하여 사용해 보는 것을 하게됩니다.

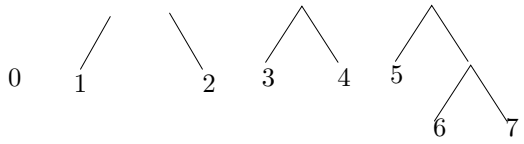
Exercise 1 "나무구조 데이타"

두 갈래 나무구조(binary tree)는 다음과 같이 귀납적으로 정의된다:

- 기본 나무구조: 잎새 하나는 나무구조이다.
- 나무구조들을 품고있는 나무구조: 하나의 꼭지에서 최대 두 개의 나무구조들이 각각 하나씩 뻗어나간 구조는 다시 나무구조이다.

위의 조건이 두 갈래 나무구조를 만드는 두가지 방법을 의미한다: 기본적인 나무를 만드는 방법(base case)과 만든 나무들을 가지고 새로운 나무를 만드는 방법(inductive case).

잎새에만 정수가 있는 두 갈래 나무구조를 생각해 보자. 예를 들어 다음의 것들이 두 갈래 나무구조들이다.



위의 구조물들은 다음과 같이 정수 혹은 짝(pair)으로 표현할 수 있다.

- 첫 번째는: 0
- 두 번째는: $(1, Nil)$

- 세 번째는: $(Nil, 2)$
- 네 번째는: $(3, 4)$
- 다섯 번째는: $(5, (6, 7))$

위와 같은 두 갈래 나무구조를 구현하는 아래 함수들을 정의하고:

$$\texttt{make\_leaf} : \mathit{int} \rightarrow \mathit{tree}$$
$$\texttt{make\_ltree} : \mathit{tree} \rightarrow \mathit{tree}$$
$$\texttt{make\_rtree} : \mathit{tree} \rightarrow \mathit{tree}$$
$$\texttt{make\_lrtree} : \mathit{tree} \times \mathit{tree} \rightarrow \mathit{tree}$$

나무를 사용하는 다음 세가지 함수들도 정의하라:

$$\texttt{is\_empty\_tree} : \mathit{tree} \rightarrow \mathit{bool}$$
$$\texttt{is\_leaf} : \mathit{tree} \rightarrow \mathit{bool}$$
$$\texttt{leaf\_val} : \mathit{tree} \rightarrow \alpha$$
$$\texttt{left\_tree} : \mathit{tree} \rightarrow \mathit{tree}$$
$$\texttt{right\_tree} : \mathit{tree} \rightarrow \mathit{tree}$$

`is_empty_tree`는 주어진 나무가 빈 나무인지를 판별한다. `is_leaf`는 주어진 나무가 잎새 나무인지 아닌지를 판별한다. `leaf_val`은 잎새가 가지고 있는 데이타를 꺼낸다. `left_tree`는 나무의 왼 편에 매달린 하부 나무를 내 놓는다. `right_tree`는 나무의 오른편에 매달린 하부 나무를 내 놓는다. □

Exercise 2 "논리회로는 나무구조"

부울 회로(Boolean circuit)라는 것은 다음과 같이 귀납적으로 정의된다. 모든 부울 회로는 하나 이상의 입력과 하나의 출력을 가지고 있다.

- 기본: 0을 가진 전기줄은 부울 회로이다.
- 기본: 1을 가진 전기줄은 부울 회로이다.
- 귀납: 부울 회로 하나를 가지고 또다른 회로를 만드는 방법 `not`이 있다.
- 귀납: 부울 회로 두개를 가지고 또다른 회로를 만드는 방법 `and`가 있다.
- 귀납: 부울 회로 두개를 가지고 또다른 회로를 만드는 방법 `or`가 있다.

부울 회로를 만드는 위의 다섯가지 방법들을 정의하라:

$$\begin{aligned}
&\texttt{zero} : \mathit{circuit}\\
&\texttt{one} : \mathit{circuit}\\
&\texttt{not} : \mathit{circuit} \to \mathit{circuit}\\
&\texttt{and} : \mathit{circuit} \times \mathit{circuit} \to \mathit{circuit}\\
&\texttt{or} : \mathit{circuit} \times \mathit{circuit} \to \mathit{circuit}
\end{aligned}$$

부울 회로가 어떻게 왜 나무구조가 되는지를 음미하자. 그리고 부울 회로를 사용하는 아래의 여섯가지 함수들도 정의하라:

$$\begin{aligned}
&\texttt{is_zero} : \mathit{circuit} \to \mathit{bool}\\
&\texttt{is_one} : \mathit{circuit} \to \mathit{bool}\\
&\texttt{is_not} : \mathit{circuit} \to \mathit{bool}\\
&\texttt{is_and} : \mathit{circuit} \to \mathit{bool}\\
&\texttt{is_or} : \mathit{circuit} \to \mathit{bool}\\
&\texttt{sub_circuit} : \mathit{circuit} \times \mathit{int} \to \mathit{circuit}
\end{aligned}$$

sub-circuit은 회로와 자연수 $n \geq 0$을 받아서 그 회로의 n번째 가지의 회로(n-th sub-circuit)를 내 놓는다. □

Exercise 3 "논리 회로의 계산"

Exercise2의 함수들로 만들어진 부울 회로의 최종 출력값(회로의 의미)를 계산하는 함수

$$\texttt{output} : \mathit{circuit} \to \{0, 1\}$$

를 정의하라.

부울 회로의 최종 출력값은 그 회로가 어떻게 만들어 졌냐에 따라 다음과 같이 재귀적으로 정의된다. zero의 출력값은 0. one의 출력값은 1. $\texttt{not}(B)$는 회로 B의 출력값이 0이면 1, 1이면 0. $\texttt{and}(B_1, B_2)$은 회로 B_1과 B_2 둘의 출력값이 모두 1일 때만 1, 아니면 0. $\texttt{or}(B_1, B_2)$은 회로 B_1과 B_2 둘의 출력값이 모두 0일 때만 0, 아니면 1. □

Exercise 4 "모빌 무게 재기"

천장에 매달려 균형을 잡은 채 은은히 흔들리고 있는 모빌을 떠올려보자. 일반적인 두갈래 모빌은 다음과 같이 정의된다:

- 모형 하나는 모빌구조이다.
- 모빌들을 품고있는 모빌: 하나의 균형점에서 왼쪽/오른쪽의 모빌구조들이 뻗어나간 구조는 다시 모빌구조이다.

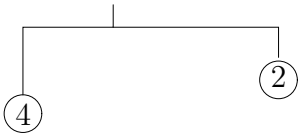
모빌구조를 만드는 다음 세 가지 함수를 정의하라:

$$\texttt{model} : \mathit{nat} \to \mathit{mobile}$$
$$\texttt{make\_branch} : \mathit{nat} \times \mathit{mobile} \to \mathit{branch}$$
$$\texttt{make\_mobile} : \mathit{branch} \times \mathit{branch} \to \mathit{mobile}$$

`model`은 입력한 자연수값만큼의 무게를 갖는 기본 모빌을 만든다.

`make_branch`는 그 길이와 끝에 매달린 모빌을 받아 하나의 가지를 이룬다.

`make_mobile`은 왼쪽/오른쪽의 가지를 받아 하나의 모빌을 이룬다.



모빌을 사용하는 다음 두 가지 함수들도 정의하라:

$$\texttt{is\_balanced} : \mathit{mobile} \to \mathit{bool}$$
$$\texttt{weight} : \mathit{mobile} \to \mathit{nat}$$

`is_balanced`는 주어진 모빌이 '균형잡혀있는지'를 판별한다. 하나의 모빌은 양쪽 가지의 토크(길이×무게)가 같을 때 균형이 잡히며, 두 가지의 무게의 합이 그 모빌의 무게가 된다. 한 모빌 구조 내의 모든 하위 모빌 구조가 균형상태에 있을 때, 그 모빌은 '균형잡혀있다'고 한다. `weight`는 그 모빌구조의 총 무게를 내놓는다. □