

# Depth-First Search

Toshiaki Nishihara      Yasuhiko Minamide

December 12, 2009

## Abstract

Depth-first search of a graph is formalized with `recdef`. It is shown that it visits all of the reachable nodes from a given list of nodes. Executable ML code of depth-first search is obtained with code generation feature of Isabelle/HOL. The formalization contains two implementations of depth-first search: one by stack and one by nested recursion. They are shown to be equivalent. The termination condition of the version with nested-recursion is shown by the method of inductive invariants.

## Contents

<b>1</b>	<b>Depth-First Search</b>	<b>1</b>
1.1	Definition of Graphs . . . . .	1
1.2	Depth-First Search with Stack . . . . .	2
1.3	Depth-First Search with Nested-Recursion . . . . .	3
1.4	Basic Properties . . . . .	3
1.5	Correctness . . . . .	4
1.6	Executable Code . . . . .	4

## 1 Depth-First Search

```
theory DFS
imports Main
begin
```

### 1.1 Definition of Graphs

```
typedcl node
types graph = (node * node) list

primrec
  nexts :: [graph, node]  $\Rightarrow$  node list
where
  nexts [] n = []
```

|  $nexts (e\#es) n = (if\ fst\ e = n\ then\ snd\ e \# nexts\ es\ n\ else\ nexts\ es\ n)$

**definition**

$nextss :: [graph, node\ list] \Rightarrow node\ set$  **where**  
 $nextss\ g\ xs = set\ g\ \text{“}\ set\ xs$

**lemma** *nexts-set*:  $y \in set (nexts\ g\ x) = ((x,y) \in set\ g)$   
(*proof*)

**lemma** *nextss-Cons*:  $nextss\ g\ (x\#xs) = set (nexts\ g\ x) \cup nextss\ g\ xs$   
(*proof*)

**definition**

$reachable :: [graph, node\ list] \Rightarrow node\ set$  **where**  
 $reachable\ g\ xs = (set\ g)^*\ \text{“}\ set\ xs$

## 1.2 Depth-First Search with Stack

**definition**

$nodes-of :: graph \Rightarrow node\ set$  **where**  
 $nodes-of\ g = set (map\ fst\ g\ @\ map\ snd\ g)$

**lemma** [*simp*]:  $x \notin nodes-of\ g \implies nexts\ g\ x = []$   
(*proof*)

**lemma** [*simp*]:  $finite (nodes-of\ g - set\ ys)$   
(*proof*)

**function**

$dfs :: graph \Rightarrow node\ list \Rightarrow node\ list \Rightarrow node\ list$

**where**

$dfs-base: dfs\ g\ []\ ys = ys$   
|  $dfs-inductive: dfs\ g\ (x\#xs)\ ys = (if\ x\ mem\ ys\ then\ dfs\ g\ xs\ ys$   
 $\quad\quad\quad else\ dfs\ g\ (nexts\ g\ x@xs)\ (x\#ys))$

(*proof*)

**declare** *wf-finite-psubset*[*simp*]

**termination**

(*proof*)

- The second argument of *dfs* is a stack of nodes that will be visited.
- The third argument of *dfs* is a list of nodes that have been visited already.

### 1.3 Depeth-First Search with Nested-Recursion

**function**

$dfs2 :: graph \Rightarrow node\ list \Rightarrow node\ list \Rightarrow node\ list$

**where**

$dfs2\ g\ []\ ys = ys$

| *dfs2-inductive:*

$dfs2\ g\ (x\#\!xs)\ ys = (if\ x\ mem\ ys\ then\ dfs2\ g\ xs\ ys$   
 $\quad\quad\quad else\ dfs2\ g\ xs\ (dfs2\ g\ (nexts\ g\ x)\ (x\#\!ys)))$

*<proof>*

**lemma** *dfs2-invariant*:  $dfs2\ dom\ (g,\ xs,\ ys) \implies set\ ys \subseteq set\ (dfs2\ g\ xs\ ys)$

*<proof>*

**termination** *dfs2*

*<proof>*

**lemma** *dfs-app*:  $dfs\ g\ (xs\@\!ys)\ zs = dfs\ g\ ys\ (dfs\ g\ xs\ zs)$

*<proof>*

**lemma** *dfs2 g xs ys = dfs g xs ys*

*<proof>*

### 1.4 Basic Properties

**lemma** *visit-subset-dfs*:  $set\ ys \subseteq set\ (dfs\ g\ xs\ ys)$

*<proof>*

**lemma** *next-subset-dfs*:  $set\ xs \subseteq set\ (dfs\ g\ xs\ ys)$

*<proof>*

**lemma** *nextss-closed-dfs* [rule-format]:

$nextss\ g\ ys \subseteq set\ xs \cup set\ ys \longrightarrow nextss\ g\ (dfs\ g\ xs\ ys) \subseteq set\ (dfs\ g\ xs\ ys)$

*<proof>*

**lemma** *nextss-closed-dfs*:  $nextss\ g\ (dfs\ g\ xs\ []) \subseteq set\ (dfs\ g\ xs\ [])$

*<proof>*

**lemma** *Image-closed-trancl*: **assumes**  $r\ \text{“}\ X \subseteq X\ \text{shows}\ r^*\ \text{“}\ X = X$

*<proof>*

**lemma** *reachable-closed-dfs*:  $reachable\ g\ xs \subseteq set\ (dfs\ g\ xs\ [])$

*<proof>*

**lemma** *reachable-nexts*:  $reachable\ g\ (nexts\ g\ x) \subseteq reachable\ g\ [x]$

*<proof>*

**lemma** *reachable-append*:  $\text{reachable } g (xs @ ys) = \text{reachable } g xs \cup \text{reachable } g ys$   
*<proof>*

**lemma** *dfs-subset-reachable-visit-nodes*:  $\text{set } (\text{dfs } g \ xs \ ys) \subseteq \text{reachable } g \ xs \cup \text{set } ys$   
*<proof>*

## 1.5 Correctness

**theorem** *dfs-eq-reachable*:  $\text{set } (\text{dfs } g \ xs \ []) = \text{reachable } g \ xs$   
*<proof>*

**theorem**  $y \in \text{set } (\text{dfs } g \ [x] \ []) \iff ((x,y) \in (\text{set } g)^*)$   
*<proof>*

## 1.6 Executable Code

**types-code**  
*node (int)*

**code-module** *DFS* **file** *DFS.sml* **contains**  
*dfs = dfs dfs2 = dfs2*

**end**