

# Depth-First Search

Toshiaki Nishihara      Yasuhiko Minamide

March 2, 2024

## Abstract

Depth-first search of a graph is formalized with `function`. 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	2
1.4	Basic Properties	3
1.5	Correctness	3
1.6	Executable Code	4

## 1 Depth-First Search

```
theory DFS
imports Main
begin
```

### 1.1 Definition of Graphs

```
typedcl node
type-synonym graph = (node * node) list

primrec nexts :: [graph, node] ⇒ node list
where
  nexts [] n = []
| nexts (e#es) n = (if fst e = n then snd e # nexts es n else nexts es n)
```



*else dfs2 g xs (dfs2 g (nexts g x) (x#ys))*

*<proof>*

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

*<proof>*

**termination** *dfs2*

*<proof>*

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

*<proof>*

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

*<proof>*

## 1.4 Basic Properties

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

*<proof>*

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

*<proof>*

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

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

*<proof>*

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

*<proof>*

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

*<proof>*

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

*<proof>*

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

*<proof>*

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

*<proof>*

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

*<proof>*

## 1.5 Correctness

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

**theorem**  $y \in set (dfs\ g\ [x]\ []) = ((x,y) \in (set\ g)^*)$   
*<proof>*

## 1.6 Executable Code

**consts** *Node* ::  $int \Rightarrow node$

**code-datatype** *Node*

**instantiation** *node* :: *equal*  
**begin**

**definition** *equal-node* ::  $node \Rightarrow node \Rightarrow bool$   
**where**  
[*code del*]: *equal-node* = *HOL.eq*

**instance** *<proof>*

**end**

**declare** [[*code abort: HOL.equal* ::  $node \Rightarrow node \Rightarrow bool$ ]]

**export-code** *dfs dfs2* in *SML file* *<dfs.ML>*

**end**