

Depth-First Search

Toshiaki Nishihara Yasuhiko Minamide

February 23, 2021

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

1	Depth-First Search	1
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	4
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)
```


$else\ dfs2\ g\ xs\ (dfs2\ g\ (nexts\ g\ x)\ (x\#\!ys))$

$\langle proof \rangle$

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

$\langle proof \rangle$

termination *dfs2*

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

1.4 Basic Properties

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

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