Connected Graphs and Spanning Trees

GAINA, Daniel

Japan Advanced Institute of Science and Technology

January 22, 2010
Describing the problem I

\[ G = (V, E) \] - graph

1. \( V \) - set of vertices
2. \( E \) - (multi)set of edges

Example:

\[ V = \{1, \ldots, 9\} \]
\[ E = \{<1,2>; <1,4>; <2,3>; <2,5>; <3,6>; <4,5>; <4,7>; <5,6>; <5,8>; <6,9>; <7,8>; <8,9>\} \]
Describing the problem II

\[ \begin{align*}
G &= (V, E) \text{ connected;} \\
T &= (V, E') \text{ spanning tree of } G \text{ when} \\
&\quad \begin{enumerate}
\item T \text{ tree,}
\item E' \subseteq E.
\end{enumerate}
\end{align*} \]

**Theorem**

*Every connected graph has a spanning tree.*
Towards formalization

1. \( \text{connected}(G) \implies \exists G' \subseteq G. \text{tree}(G') \)

2. \( \text{connected}(G) \implies \text{connected}(\text{mktree}(G)) \land \text{nocycle}(\text{mktree}(G)) \)

To do:

1. data representations for mathematical objects (graphs);
2. define
   - connected
   - nocycle
   - mktree
Functions on graphs

\[ G = (\{1, 2, 3, 4, 5, 6, 7, 8\}, < 1, 2 >; < 1, 6 >; < 1, 5 >; < 3, 4 >; < 3, 7 >; < 4, 7 >; < 5, 6 >) \]

\[ \text{mcc}(A, G) = \text{max. connected comp. of } A \text{ in } G. \]

\[ \text{mcc}(6, G) = \{1, 2, 5, 6\}, \text{ mcc}(8, G) = \{8\} \]

\[ \#cc(G) = \text{no. of max. connected components} \]

\[ \#cc(G) = 3 \]

\[ \text{nocycle}(G) = false \]
Spanning forests I

in the attempt of proving the desired properties we realized is much easier to prove a more general result:

**Every graph has a spanning forest!**

**Definition**

A **spanning forest** of a graph is a subgraph that consists of a set of spanning trees, one for each maximal connected component of the initial graph.

We define the function `mktree` which returns the spanning forest of a graph.
Spanning forests II

\[ G = (\{1, 2, 3, 4, 5, 6, 7, 8\}, \langle 1, 2 \rangle; \langle 1, 6 \rangle; \langle 1, 5 \rangle; \langle 3, 4 \rangle; \langle 3, 7 \rangle; \langle 4, 7 \rangle; \langle 5, 6 \rangle) \]

\[ \text{mktree}(G) = (\{1, 2, 3, 4, 5, 6, 7, 8\}, \langle 1, 2 \rangle; \langle 1, 6 \rangle; \langle 1, 5 \rangle; \langle 3, 4 \rangle; \langle 3, 7 \rangle) \]

Remark

The value is relative to the order chosen for the edges.
Properties to be proved

Assuming that we have defined

- $\text{mcc}$, $\#cc$, $\text{nocycle}$, $\text{mktree}$

we need to prove

1. $\text{mcc}(A, G) = \text{mcc}(A, \text{mktree}(G))$
2. $\#cc(G) = \#cc(\text{mktree}(G))$
3. $\text{nocycle}(\text{mktree}(G))$

Then we define $\text{connected}(G) := (\#cc(G) = 1)$ which implies

- $\text{connected}(G) \Rightarrow \text{connected}(\text{mktree}(G)) \land \text{nocycle}(\text{mktree}(G))$
Set I

mod* ID {
[Id]
- equality on Id
op _=_ : Id Id -> Bool {comm}
vars I J : Id
eq [i1] : (I = I) = true .
ceq [i2] : I = J if (I = J) .
}

mod* SET(I :: ID) {
[Id < Set]
op empty : -> Set {constr}
op (_U_) : Set Set -> Set {constr assoc comm}
eq (S:Set U S) = S .
vars I I’ : Id
vars S S’ : Set
Set II

- $(I \text{ in } S)$ indicates whether $I$ is an element of $S$ or not
  
  $\text{op } _\text{in}_ : \text{ Id Set } \rightarrow \text{ Bool }.$

  $\text{eq } I \text{ in } \emptyset = \text{ false }.$

  $\text{eq } I \text{ in } I' = \text{ if } I = I' \text{ then true else false fi }.$

  $\text{eq } I \text{ in } I' \cup S = \text{ if } I = I' \text{ then true else } I \text{ in } S \text{ fi }.$

- $(S <s S')$ indicates whether $S$ is subset of $S'$
  
  $\text{op } _\text{<s}_ : \text{ Set Set } \rightarrow \text{ Bool }.$

  $\text{eq } \emptyset <s S = \text{ true }.$

  $\text{eq } I <s S = \text{ if } I \in S \text{ then true else false fi }.$

  $\text{eq } (I \cup S) <s S' = \text{ if } I \in S' \text{ then } (S <s S') \text{ else false fi }.$
Set III

- equality on Set
  
op _=_: Set Set -> Bool \{comm\}
  
eq [s1]: (S = S) = true .
  
eq [s2]: (S = S') = (S <s S') and (S' <s S) .
  
ceq [s3]: S = S' if (S = S') .

mod* VERTEX {
  [Vertex]
  op _=_ : Vertex Vertex -> Bool {comm}
  vars A B : Vertex
  eq [v1] : (A = A) = true .
  ceq [v2] : A = B if (A = B) .
}

mod* GRAPH(V :: VERTEX){
  [Edge]
  [Graph]
  op <_,_> : Vertex Vertex -> Edge {constr}
  op nil : -> Graph {constr}
  op _;_ : Edge Graph -> Graph {constr}
}
Remark

- Edge and Graph are constrained.
- Models consist of interpretations of terms formed with constructor and elements of sort Vertex.
mod* SFOREST (V :: VERTEX){
inc(INT)
inc(SET(V{sort Id -> Vertex})*{sort Set -> VtxSet})
inc(GRAPH(V))

vars A B C : Vertex
var G : Graph

- mcc(A,G) = max. connected component of A in G
op mcc : Vertex Graph -> VtxSet
eq mcc(A,nil) = A .
eq mcc(A, < B,C > ; G) =
  if mcc(A,G) = mcc(B,G) or mcc(A,G) = mcc(C,G)
  then (mcc(B,G) U mcc(C,G)) else mcc(A,G) fi .
SFOREST II

op nocycle : Graph -> Bool

eq nocycle(nil) = true.

eq nocycle(< A,B > ; G) =
  if mcc(A,G) = mcc(B,G) then false
  else nocycle(G) fi.

- #cc(G) = no. of max. connected comp. of G

op #cc : Graph -> Int.

op #vertices : -> Nat.

eq #cc(nil) = #vertices. - no. of vertices

eq #cc(< A,B > ; G) = if mcc(A,G) = mcc(B,G) then
  #cc(G) else #cc(G) - 1 fi.
- \texttt{mktree(G)} returns the spanning forest of \textit{G}.

\begin{verbatim}
op mktree : Graph -> Graph
eq mktree(nil) = nil .
eq mktree(< A,B > ; G) =
    if \texttt{mcc(A,G)} = \texttt{mcc(B,G)} then mktree(G)
    else < A,B > ; mktree(G) fi .
\end{verbatim}
Properties to be proved

Theorem

\textit{\texttt{mktree}(G) is a spanning forest of } G. \\

\begin{enumerate}
\item \forall G. \forall A. \texttt{mcc}(A, \texttt{mktree}(G)) = \texttt{mcc}(A, G)
\item \forall G. \#\texttt{cc}(\texttt{mktree}(G)) = \#\texttt{cc}(G)
\item \forall G. \texttt{nocycle}(\texttt{mktree}(G))
\end{enumerate}
First Theorem I

**Lemma**

Max. connected comp. of $G$ are the same as max.connected comp. of $\text{mktree}(G)$ i.e.

$\forall G. \forall A. \text{mcc}(A, \text{mktree}(G)) = \text{mcc}(A, G)$

Proof by induction on the structure of $G$.

**IB** $\forall A. \text{mcc}(A, \text{mktree}(\text{nil})) = \text{mcc}(A, \text{nil})$

**IS** $\forall G. \forall A'. \text{mcc}(A', \text{mktree}(G)) = \text{mcc}(A', G) \Rightarrow$

$\forall A. \forall B. \forall C. \text{mcc}(A, \text{mktree}(<B, C>; G)) = \text{mcc}(A, <B, C>; G)$
First Theorem II

For the induction base

open SFOREST
op a : -> Vertex .
red mcc(a,mktree(nil)) = mcc(a,nil) .
close

For the induction step

ops a b c : -> Vertex .
op g : -> Graph .
eq [IH] : mcc(A:Vertex,mktree(g)) = mcc(A,g) .
- equations corresponding to each subcase ...
red mcc(a,mktree(< b,c > ; g)) = mcc(a, < b,c > ; g) .
First Theorem III

Equations corresponding to each subcase

1. \( \text{eq } \text{mcc}(b,g) = \text{mcc}(c,g) \)
   \( \text{eq } \text{mcc}(a,g) = \text{mcc}(c,g) \)

2. \( \text{eq } \text{mcc}(b,g) = \text{mcc}(c,g) \cdot \)
   \( \text{eq } (\text{mcc}(a,g) = \text{mcc}(c,g)) = \text{false} \cdot \)

3. \( \text{eq } (\text{mcc}(b,g) = \text{mcc}(c,g)) = \text{false} \cdot \)
   \( \text{eq } \text{mcc}(a,g) = \text{mcc}(b,g) \cdot \)

4. \( \text{eq } (\text{mcc}(b,g) = \text{mcc}(c,g)) = \text{false} \cdot \)
   \( \text{eq } \text{mcc}(a,g) = \text{mcc}(c,g) \cdot \)

5. \( \text{eq } (\text{mcc}(b,g) = \text{mcc}(c,g)) = \text{false} \cdot \)
   \( \text{eq } (\text{mcc}(a,g) = \text{mcc}(b,g)) = \text{false} \cdot \)
   \( \text{eq } (\text{mcc}(a,g) = \text{mcc}(c,g)) = \text{false} \cdot \)
Second Theorem I

Theorem

\[ \text{mktree preserves the number of maximal connected components, i.e. } \forall G. \#cc(\text{mktree}(G)) = \#cc(G). \]

Proof by induction on the structure of \( G \).

1B \( \#cc(\text{mktree}(\text{nil})) = \#cc(\text{nil}) \)

1S \( \forall G. \#cc(\text{mktree}(G)) = \#cc(G) \Rightarrow \forall B. \forall C. \#cc(\text{mktree}(<B,C>;G)) = \#cc(<B,C>;G) \)
Second Theorem II

For the induction base

open SFOREST + EQL
red \#cc(mktree(nil)) = \#cc(nil) .
close

For the induction step

open SFOREST + EQL
ops a b : -> Vertex .
op g : -> Graph .
eq [IH] : \#cc(mktree(g)) = \#cc(g) .

\begin{enumerate}
\item eq mcc(a,g) = mcc(b,g) .
\item eq (mcc(a,g) = mcc(b,g)) = false .
\end{enumerate}
red \#cc(mktree(< a,b > ; g)) = \#cc(< a,b > ; g) .
Third theorem I

Theorem

\texttt{mktree}(G) \textit{has no cycles, i.e.} \\
\forall G. \texttt{nocycle}(\texttt{mktree}(G)) = \text{true}.

Proof by induction on the structure of \texttt{G}.

**IB** \texttt{nocycle}(\texttt{mktree}(\texttt{nil})) = \text{true}

**IS** \forall G. \texttt{nocycle}(\texttt{mktree}(G)) = \text{true} \Rightarrow \\
\forall B. \forall C. \texttt{nocycle}(\texttt{mktree}(\langle B, C \rangle; G)) = \text{true}
Third theorem II

For the induction base

open SFOREST
red nocycle(mktree(nil)) .
close

For the induction step

open SFOREST
ops a b : -> Vertex .
op g : -> Graph .
eq [IH] : nocycle(mktree(g)) = true .

1. eq mcc(a,g) = mcc(b,g) .
2. eq (mcc(a,g) = mcc(b,g)) = false .

red nocycle(mktree(< a,b > ; g)) .
Conclusions

- we have proved a more general property (e.g. every graph has a spanning forest) in order to achieve our goal;
- we didn’t use initial semantics;
- constructor-based logics sufficient for verifications;
- the data structure $\text{VERTEX}$ for the set of vertices is very general and can be instantiated with natural numbers;
Exercise

1. Prove $\forall G. \forall A. \forall B. (<A, B> \text{ in } G) \text{ if } (<A, B> \text{ in } \text{mktree}(G))$

2. A path between the vertex $A$ and vertex $B$ is a sequence of edges $<A_1, B_1> \ldots <A_n, B_n>$ such that:
   1. $A_1 = A$
   2. $B_n = B$ and
   3. $A_{i+1} = B_i$ for all $i \in \{1, \ldots, n-1\}$.

   A cycle is a path $<A_1, B_1> \ldots <A_n, B_n>$ such that $A_1 = B_n$

   Prove that if there exists a path between $A$ and $B$ then there exists a path with nocycles