Constructing a subspanning palm in an undirected graph.

Let G be an undirected connected graph with V as its set of vertices and E as its set of edges. We assume G to be such that it has no cycles of length 1 .
A subspanning palm of G is a rooted subspanning tree of G such that each edge of G connects two vertices the one of which is on the root path of the other.
Traditionally, such a subspanning palm emerges if the graph G is submitted to the so-called "Depth First Search", a famous algorithm, invented by R.E. Tarjan [0], with many intriguing applications.

The purpose of this essay is threefold.
Firstly, it is felt that the algorithm is so beautiful that it deserves a beautiful modern description, i.e. a description more rigorous and less operational than we are used to.
Secondly, it is felt that in the traditional treatment the cart is put before the horse: the really interesting aspect of the Depth First Search is not the algorithm itself but its outcome, the subspanning palm, and we would prefer to see how the algorithm emerges from the desire to construct a subspanning palm.
Thirdly, both the algorithm and the data involved are sufficiently complicated to serve as a challenge to undertake a formal treatment which, to the best of our knowledge, has not been given thus far.

*

Whenever we want to give a formal account of a program's correctness, it is obligatory to characterize, precisely and concisely, the net-effect to be accomplished by the program.
In the current example the notion of a rooted tree plays such a central role that a formal definition of such a tree is the very first thing we are heading for.
Typical of a rooted tree is that

a) there is a root --let us call it r --
b) each node u distinct from r has a unique predecessor node --let us call it f(u) --
c) each node u distinct from r has a unique connection to r --its root path-- i.e., for each node u : $(\underline{E}i: i \geq 1: f^{(i)}(u) = r)$.

The function f thus introduced is a partial function since it is not defined on r. Wishing to avoid partial functions like the plague --they easily lead to a diffusing case-analysis-- we rashly continuate f so as to make it a total function. Inspired by typicality c) we suggest that for the chosen r : $f(r) = r$.

Another notion of central importance is the notion of a node v lying on the root path of another node u. We propose to characterize this circumstance by the truth of the predicate

$$ff(u,v): \quad (\underline{E}i: i \geq 0: f^{(i)}(u) = v) .$$

As an abbreviation we introduce the auxiliary predicate

$$fff(u,v): \quad ff(u,v) \ \underline{or} \ ff(v,u) ,$$

expressing that v is on the root path of u or the other way around.

Using the introduced terminology, we regard it as our task to compute a pair (r,f) satisfying

R: Ra and Rb and Rc and Rd

where

Ra: $r \in V$ and $f: V \rightarrow V$ and $f(r) = r$

Rb: $(\underline{A}u: u \in V: (u,f(u)) \in E)$

Rc: $(\underline{A}u: u \in V: ff(u,r))$

Rd: $(\underline{A}(u,v): (u,v) \in E: fff(u,v))$.

The first three conditions express that the pair (r,f) forms a rooted subspanning tree of G, and the fourth condition says that the tree is even a palm.

So far for the formal characterization of the intended net-effect.

+

One of the standard techniques that enables us establish R is to depart from a relation P that is obtained from R by replacing some constants in R by variables from a suitable domain. The only two constants being V and E we therefore consider

P: Pa and Pb and Pc and Pd and Pe

in which

Pa: $r \in sv$ and $f: sv \to sv$ and $f(r) = r$

Pb: $(\underline{A}u: u \in sv: (u, f(u)) \in se)$

Pc: $(\underline{A}u: u \in sv: ff(u,r))$

Pd: $(\underline{A}(u,v): (u,v) \in se: fff(u,v))$

Pe: $sv \subset V$ and $se \subset E$ and $(\underline{A}(u,v): (u,v) \in se: u \in sv$ and $v \notin sv)$.

(Pe originates from usual domain restrictions: it expresses that the graph with sv as the set of its vertices and with se as the set of its edges forms a subgraph of the original graph G .)

Then, clearly, $(P$ and $se = E) \Rightarrow R$, so that it is tempting to investigate a program of the form

```
r:= "a vertex of V "; f:(r) = r; sv, se := [r], [(r,r)];
do se ≠ E → "transmit an edge from E - se to se
            under invariance of P "
od.
```

(In order to establish Pb upon initialisation we need the nonexisting edge (r,r), the presence of which we now postulate.)

+

Notational remark: For the sake of convenience we shall introduce two auxiliary variables svc and sec to denote sv 's complement in V and se 's complement in E respectively. If a pair of vertices u and v is connected by an edge of E , we denote this fact by writing the edge as (u,v) . If we write $(u,v) \notin$ sec we therefore mean $(u,v) \in$ se .
(End of notational remark.)

Thus we are led to consider the transmission of an edge (x,y) from sec to se . The required invariance of Pe tells us that for that edge we have to establish $x \in sv$ and $y \in sv$.
In order to avoid abrupt and bulky changes of the tree, i.e. in order to keep our arguments as simple as possible, we shall restrict ourselves to

- selections of edges (x,y) such that $y \in sv$: the connectedness of the graph G guarantees that this is always possible;
- tree adjustments that take the form of extensions $f:(x) = y$ only.

With these self-imposed constraints we have gained clarity to the extent that we have hardly any other choice than considering as a refinement of

> "transmit an edge from $E - se$ to se under invariance of P ":
>
> "select an edge (x,y): $y \in sv$ from sec ";
> $\underline{if}\ x \notin sv \rightarrow f:(x) = y;\ sv, se := sv + [x],\ se + [(x,y)]$
> $[]\ x \in sv \rightarrow se := se + [(x,y)]$
> $\underline{fi}$.

As it stands this text does not maintain P . But it does maintain Pa $\underline{and}$ Pb $\underline{and}$ Pc $\underline{and}$ Pe , which is easily verified. It means that the program we have is good enough for the construction of an arbitrary subspanning tree of G , which in the light of our task to construct a palm is a comforting observation.

The first alternative maintains Pd as well. The second alternative, however, can destroy it. It does destroy Pd if for the edge (x,y) selected $\underline{non}$ $fff(x,y)$ holds. In terms of the tree and the graph this violation means that the tree has attained a shape such that the graph provides a connection between two tree nodes none of which is on the root path of the other. If we want to preclude such an ineffective and undesired growth of the tree, our almost exclusive hope is that we can find a suitable relationship between the tree and the rest of the graph.

An edge of sec connecting two nodes of sv is a special instance of an "off-tree" path connecting them.
We define, for two arbitrary nodes u and v ,

$$off(u,v): \quad (u,v) \in sec\ \underline{or}\ (\underline{E}w: w \in svc: off(u,w)\ \underline{and}\ off(w,v)) .$$

Then, the second alternative does not violate Pd anymore if we impose the extra condition

$$H: \quad (\underline{A}u,v: u \in sv,\ v \in sv: fff(u,v)\ \underline{or}\ \underline{non}\ off(u,v)),$$

because the edge selected from sec is such that $x \in sv$ $\underline{and}$ $y \in sv$, so that $off(x,y)$ yields the value true, so that $fff(x,y)$ is satisfied.

So, let us book our interest in the maintenance of H .
Extensions of the sets sv and se respectively can never violate the falsity of off(u,v) . Hence, the only possibility to falsify H is by a tree extension f:(x) = y whenever it so happens that an off-tree path exists between x and a node of the tree which is not on the root path of x .
If we wish to maintain H at a bargain, we have to ensure that all off-tree paths connecting x with the tree connect x with nodes on its own root path. This imposes a strong limitation on the freedom we have in selecting the edge (x,y) : the property x has to enjoy should follow more or less directly from a similar property of y . This means, that we possibly can gain efficiency in the subsequent selections of edges if we introduce, instead of H , a stronger relation Q(y) which mentions y .
If the maintenance of Q(y) for the sake of gaining efficiency is to make sense at all, we must restrict ourselves to easy adjustments of y . Besides the strongly suggested y:= x the only other simple adjustment of y is y:= f(y) , pushing y "rootwards". But then, the only tree nodes accessible from y are those on the root path of y , and Q(y) will have to be more than just a strengthening of H because it should also express the possibility to select an edge adjacent with the root path of y .
Some walking up and down reveals that both requirements on Q(y) can be gracefully combined by the definition

Q(y): (Au,v: u ∈ sv, v ∈ V: ff(y,u) or non off(u,v)) ,

reading: all off-tree connections between a node of the tree and another node have an endpoint on the root path of y .
(The fact that Q(y) implies H is easily seen by assuming the truth of off(u,v) for a pair of nodes u and v from sv : Q(y) then infers ff(y,u) and ff(y,v) which implies fff(u,v) .)

+

So far for the discovery of what might be relevant. We shall now prove that we indeed can come away with a computation maintaining the relation P and Q(y) . Our remaining obligations consist of showing

A: $\{P \text{ and } Q(y) \text{ and } x \notin sv \text{ and } y \in sv \text{ and } (x,y) \in sec\}$

$f:(x) = y;\ sv, se := sv + [x], se + [(x,y)];\ y := x$

$\{Q(y)\}$

and

B: $\{P \text{ and } Q(y) \text{ and } y \in sv \text{ and } ec(y) = \emptyset\}\quad y := f(y)\quad \{Q(y)\}$,

in which $ec(y)$ is the set of edges from sec which are adjacent to y .

Proof of A:

- $\{P \text{ and } Q(y) \text{ and } x \notin sv \text{ and } y \in sv \text{ and } (x,y) \in sec\}$

  $f:(x) = y$

  $\{Q(x) \text{ and } (\underline{A}v: v \in V: ff(x,x) \text{ or non } off(x,v))\}$

  -- the first factor in the post-condition follows from the fact that $(ff(y,u) \text{ and } f(x) = y) \Rightarrow ff(x,u)$;

  -- the second factor follows from the universal truth of $ff(x,x)$;

- $\{Q(x) \text{ and } (\underline{A}v: v \in V: ff(x,x) \text{ or non } off(x,v))\}$

  $sv, se := sv + [x], se + [(x,y)]$

  $\{Q(x)\}$

  following from simple rewriting rules and from the fact that extensions of the sets sv and se respectively preserve the falsity of $off(u,v)$;

- $\{Q(x)\}\quad y := x\quad \{Q(y)\}$

so that the Rule of Composition can do the rest.

(End of proof of A.)

Proof of B:

- $(Q(y) \text{ and } y \in sv \text{ and } ec(y) = \emptyset \text{ and } u \in sv \text{ and } off(u,v)) \Rightarrow (ff(y,u) \text{ and } u \neq y)$;

- $(ff(y,u) \text{ and } u \neq y) \Rightarrow ff(f(y),u)$,

so that the Axiom of Assignment can do the rest.

(End of proof of B.)

Program:

```
r:= "a vertex of  V "; f:(r) = r; sv, se := [r], [(r,r)]; y:= r;
do se ≠ E → do  ec(y) = ∅ → y:=f(y) od;
            "select an edge  (x,y)  from  ec(y) "; se:= se + [(x,y)];
            if x ∉ sv → f:(x) = y; sv:= sv + [x]; y:= x
             [] x ∈ sv → skip
            fi
od.
```

The connectedness of the graph G together with $se \neq E$ ensures the existence of a node u: $u \in sv$ such that $ec(u) \neq \emptyset$, which in combination with Q(y) ensures the existence of an integer i: $i \geq 0$ such that $ec(f^{(i)}(y)) \neq \emptyset$, so that the inner repetitive construct terminates.

All sorts of embellishments like strengthening the guard of the outer repetitive construct into $sv \neq V$ and like the development of real code, fall --because they are easy-- outside the scope of this manuscript.

*

Final remarks:

We have done rather extensive experiments with the problem of constructing a subspanning palm. Two earlier texts have disappeared into the waste-paper basket. The first main difficulty was to clean the mind by trying to forget that we knew the Depth First Search almost by heart. The second difficulty --which was of a more technical nature-- was the discovery of the relation called Q(y) . It emerged only after the decision was taken to restrict the domain of the node named y to the nodes of its own root path. From that moment onwards the Depth First Search considered as an algorithm to determine a subspanning palm is uniquely induced.

In the earlier versions of the above text we tried to be very pure --poor, by now-- by ignoring completely the nomenclature as offered by graph theory, such as the notions of a tree, a root path, off-tree paths, etc.. We tried to drive the whole development by exploring properties and structure of formal objects. Although the approach appeared to be possible, it became very cumbersome, to put it mildly: the formal game got oversized and was, perhaps therefore, a

poor source of inspiration. In the current text we have not eschewed the metaphor of the tree and the graph, thus providing ourselves with a welcome mental aid, however without basing our reasoning upon them. As such, the experiments served to rediscover the rule that in developing programs the role of well-chosen metaphors and a nomenclature for them can be of indispensable value, and that reasoning by formulae manipulation alone is a poor substitute for good thinking. This definitely is a drawback for all sorts of efforts aiming at automatic program verification.

As was to be expected, the Depth First Search as captured by the trivial P and by the nice and concise Q(y) is better understood than ever before, at least in the author's mind.
(End of final remarks.)

[0] R.E. Tarjan, Depth-first search and linear graph algorithms, SIAM J. Comput. 1(1972) 146-160 .

W.H.J. Feijen,
Eindhoven, July 30, 1980 .