Bourdoncle Components

Matt Elder
elder@cs.wisc.edu

28 June 2010

Abstract

This is a very high-level overview of weak topological ordering (WTO), and Bourdoncle's algorithm for finding WTOs. Weak topological ordering is a mechanism used in abstract interpretation to find CFG vertices suitable for widening, and to direct interpretation strategies.

1 Motivation

Many abstract domains require widening. It helps both precision and efficiency to widen at as few nodes as possible. If we have a vertex ordering in mind, then it is safe to widen just on the targets of *feedback edges* - edges of the graph that go backwards in our ordering. (This is a rigorous version of the intuitive strategy of matching at the heads of loops.)

To minimize the number of widening points, we would like to select our vertex ordering in order to minimize the number of targets of feedback edges. This is the *minimal feedback vertex set* problem, known to be NP-complete [2]. Since we cannot expect an efficient and correct algorithm, we'll settle for an algorithm that works well in practice.

2 Weak Topological Ordering

Suppose we have an ordering of the vertices of a graph, with well-matched parentheses added. We define:

- A *component* (or *Bourdoncle component*) is the set of vertices inside a matched pair of parentheses.
- A *head* is the leftmost element of a component.
- A *feedback edge* is an edge of the graph that goes backwards in the ordering.
- The *height* of a vertex is the number of pairs of parentheses that contain in. The height of the ordering is the maximum height of its elements.

A *weak topological ordering* (or *Bourdoncle ordering*, or *WTO*) is a well-parenthesized ordering of the vertices of a directed graph in which:

- No two left parentheses are adjacent; thus, the head of every component is in no subcomponent.

- If $u \to v$ is a feedback edge, then v is the head of some component containing u.

For our purposes, we prefer WTOs to have as few components as possible, and for the size and depth of each component to be as small as possible. Thus, in the example graph of Figure 1:

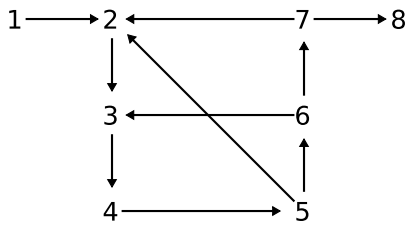

Figure 1: Example graph

- 1 (2 (3 4 5 6) 7) 8 is a WTO.
- 1 (2 (3 4 5 6 7 8)) is a WTO, but is less desirable.
- 1 (2 (3 (4 (5) 6)) 7) 8 is a WTO, but is embarrassingly bad.
- 1 (2 (3 4 5) 6 7) 8 is not a WTO, because $6 \to 3$ is a feedback edge of the ordering but 6 isn't in the component that 3 is the head of.

3 Widening strategy

If we have a WTO of a CFG, then we can use the set of heads of the ordering as widening points for abstract interpretation. Moreover, the nesting of the components gives us a good fixpoint strategy: Bourdoncle's recursive strategy of [1] handles each component as an "iterate until stabilization" operator. Suppose that 1 (2 (3 4) 5 6) 7 is a WTO of a CFG. Bourdoncle's recursive strategy does this:

```
Interpret 1
repeat
    Interpret 2
    repeat
        Interpret 3
        Interpret 4
    until values stabilize
    Interpret 5
    Interpret 6
until values stabilize
Interpret 7
```

4 WTO Algorithm

At a very high level, Bourdoncle's algorithm for finding WTOs from a directed graph G is as follows:

1: **function** WTO(G)
2: Compute $\mathcal{C}$, the set of strongly-connected components of G.
3: **for all** $C \in \mathcal{C}$ **do**
4: **if** $|C| = 1$ **then**
5: leave C as-is.
6: **else**
7: Select some $h \in C$.
8: Replace C with $Component(h, \text{WTO}(C \setminus \{h\})$.
9: **end if**
10: **end for**
11: Write the resulting singletons and components in topological order.
12: **end function**

In the above, $Component$ is a data constructor that holds an ordered list of vertices and components.

To emphasize: h at Line 7 is selected heuristically; this is where the algorithm trades optimality for efficiency. Any selection rule at Line 7 will yield a correct WTO. Careful heuristic selection is required, though, to yield a good WTO.

In [1], Bourdoncle's algorithm finds strongly-connected components (Line 2) via Tarjan's algorithm. [3] It selects h at Line 7 as the first element of C visited by the depth-first search that Tarjan's algorithm performs. Its topological ordering (Line 11) is implicit in the ordering that Tarjan's algorithm returns components.

References

[1] F. Bourdoncle. Efficient chaotic iteration strategies with widenings. In Formal Methods in Programming and Their Applications, 1993.

[2] M. R. Garey and David S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979

[3] Robert Tarjan. *Depth-first search and linear graph algorithms*. In: SIAM Journal on Computing. Vol. 1, 1972.