Static over-approximation for (mutually) recursive functions in the construction of the abstract event graph

In the case of recursive calls, we can connect the abstract events of the function so that any cycle appearing in a sequence of dynamic calls can appear in the abstract event graph as a cycle. The strategy is similar to the method used for unbounded loops. In this construction, we call *last events of the body* of a function the events that are preceding any return in the function. The *first events of the body* of a function are the first events of the subgraph built out of the function.

1. The function calls are first "unfolded twice": we explore the body of the recursive function with a depth of 2, as if we were inlining them twice;
2. We construct inside the body the abstract event graph as before;
3. At the recursive callsite, we connect by po_s the last abstract events of the first function body before the call to the first abstract events of the second body, and the last abstract events of the second body to the first abstract events after the call in the first body. This models the simplest dynamic call case, in which an event of the caller and an event of the callee could both belong to a critical cycle;
4. Then we connect by po_s the last abstract events before the recursive call in the second body to the first abstract events in the first body, and we connect the last abstract events of the first body to the first abstract events after the recursive call in the second body. This models the recursive calls, in which one might encounter a critical cycle involving twice the same static abstract event.

We construct the cmp during this process as before. Any cycle occurring for a sequence of dynamic function calls is a cycle in this subgraph. To extend this to mutually recursive functions, we take the *period* in the sequence of recursive calls to the first function, that is, the smallest sequence of calls that is repeated, then duplicate it and apply the same transformations, as depicted in Fig. 1.

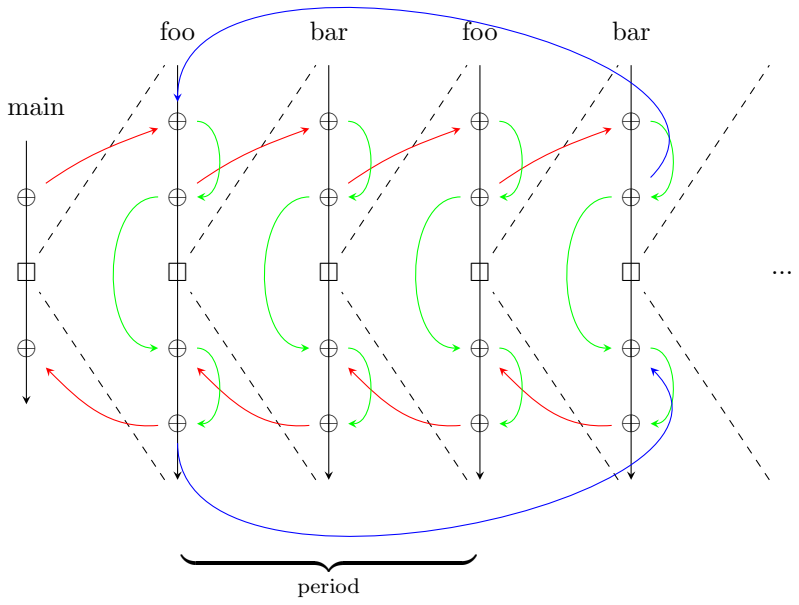

Figure 1: Construction of po_s for mutual recursive functions. Each of the vertical line is the body of a function. The circles are abstract events, the squares are call-sites, the dashed lines represent a potential function call. The coloured edges are the po_s edges we construct: in green, the edges built in step 2; in red, the edges added in step 3; in blue, the edges described in step 4. Note that the absence of a green edge over a recursive callsite would mean that this function would trivially not terminate if the callsite is reachable.