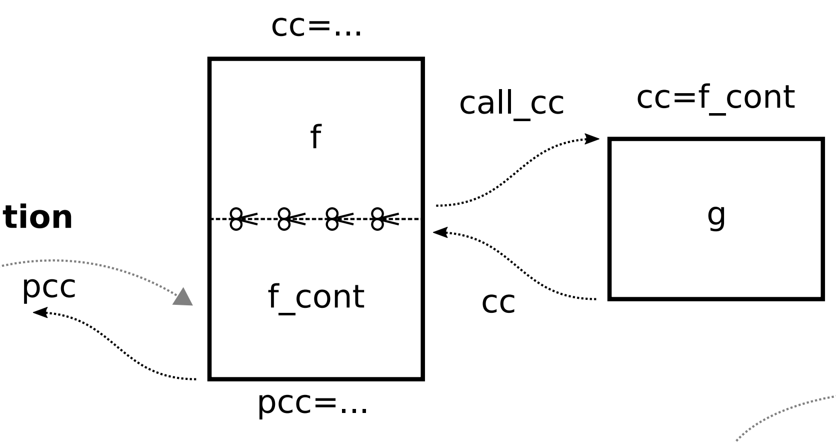
call_cc: running [code] with scissors

Base case: One continuation

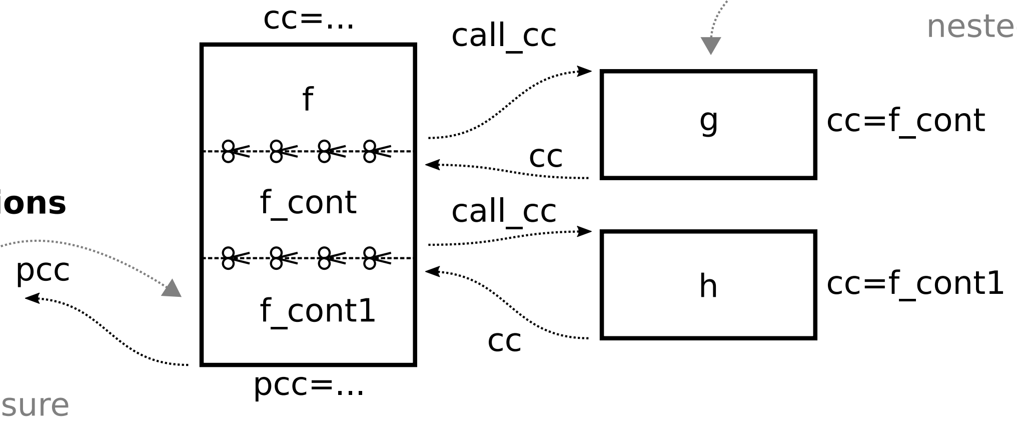


g can stash cc for use later (that's the whole point)

f_cont is a closure that is instantiated when f is called; it lives in the lexical scope of f.

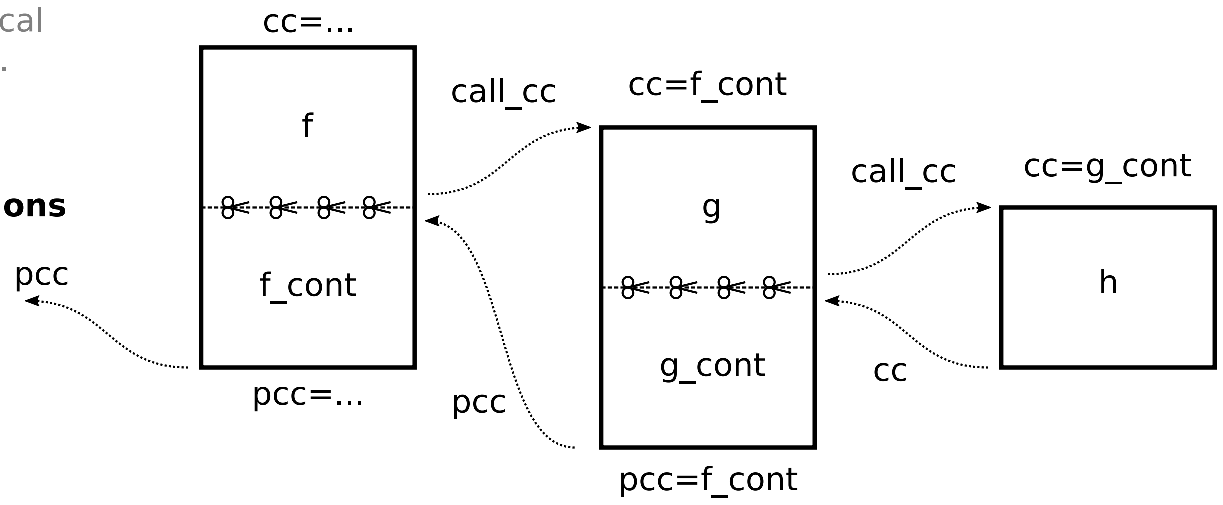
To see where this usage pattern is useful, think g=h=amb ⇒ can behave like nested loops

Sequence of continuations



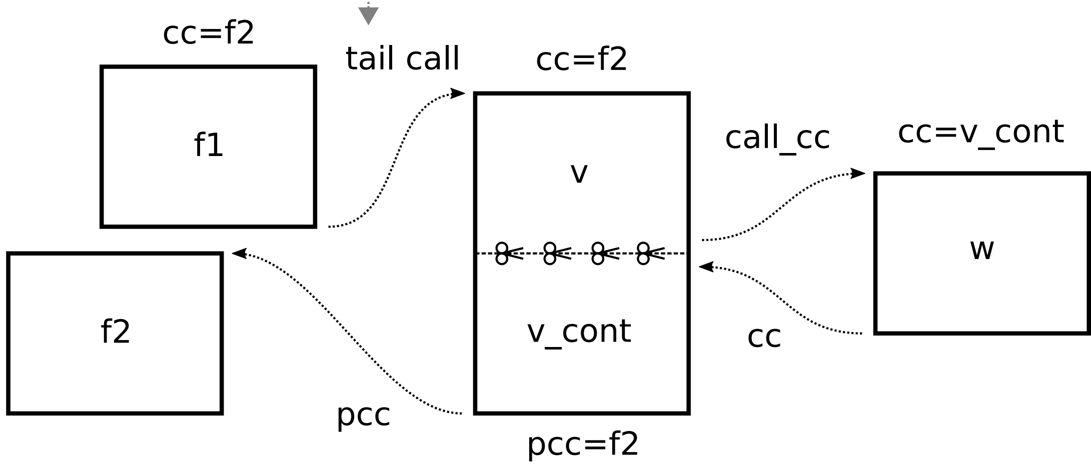
The f_cont1 closure lives in the lexical scope of f_cont.

Nested continuations

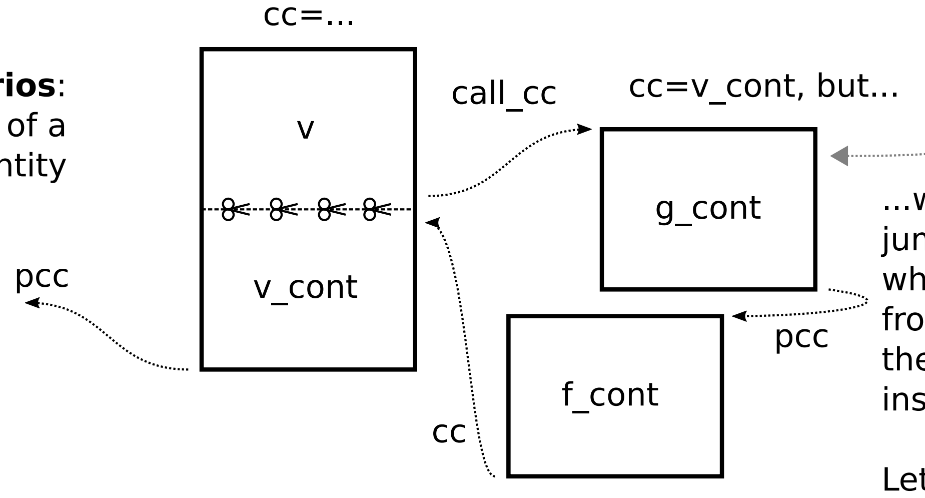


A tail call must propagate the value cc has at the site of the tail call.

Nearly equivalent?



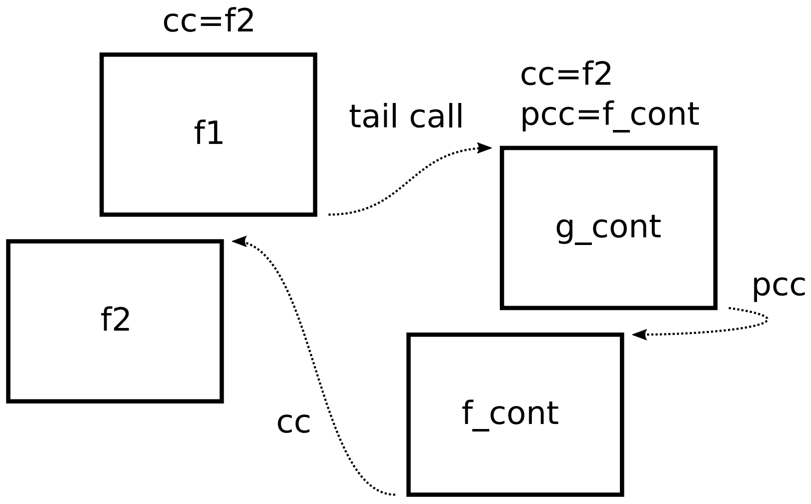
The Confetti Scenarios: pretending the whole tail of a computation is just one entity

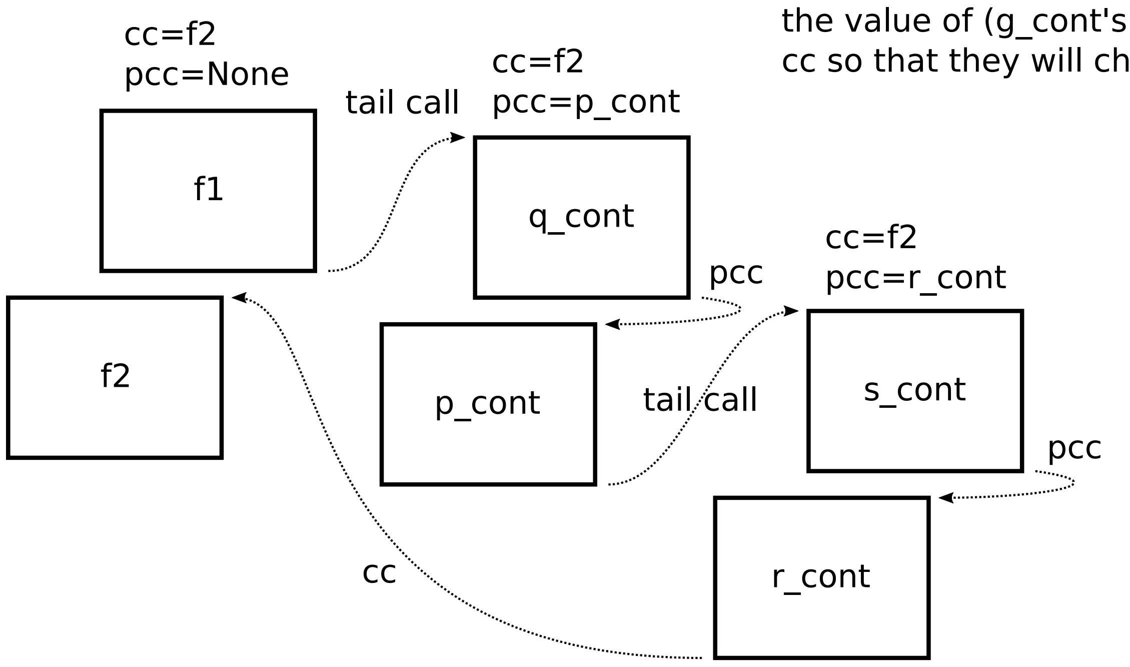


Stashed earlier by h (above)

...we should first jump to f_cont, which is the cc of g from the time when the g_cont closure instance was created.

Let's call it **pcc** for *parent cc*, and when calling it, pass to it the value of (g_cont's) cc so that they will chain.





When a function ends, check for pcc first; if it's set, tail-call it (and set its cc); if it isn't, then tail-call the cc.

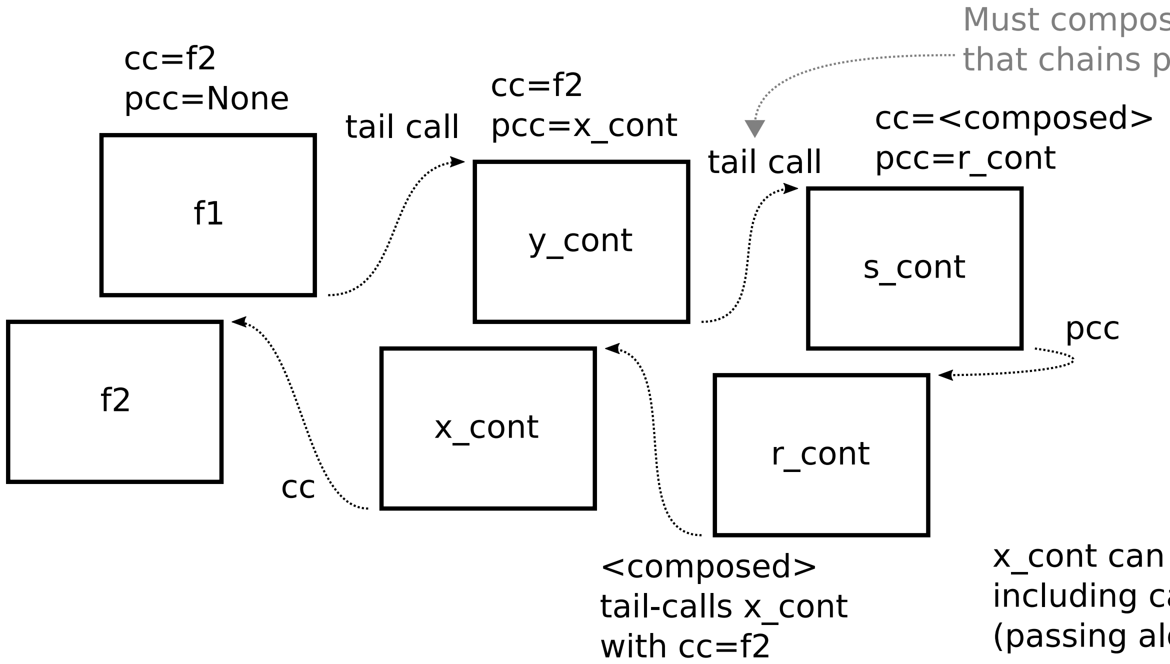
A separate arg is needed for the pcc, because the *cc* arg is a public API, for setting by call_cc and tail calls.

The *cc* arg stores nothing persistently, so we may use the *cc* arg of pcc to pass in the desired continuation.

This also chains correctly if the tail consists of three or more parts (e.g. h_cont, g_cont, f_cont): the cc that was passed in will be invoked **last**, after the pcc chain itself completes.

The only place that sets *pcc* is the call_cc mechanism that creates the definition of the continuation function.

Finally, we need one more mechanism to treat the case where both cc and pcc are set, and a tail call is encountered:



Must compose and pass along a cc that chains pcc and cc, in that order.

So the general solution for a tail call is to check for pcc; if set, make a composed cc; if not, just pass along the existing cc.

This chains correctly also in the presence of more nested tail calls.

x_cont can internally do whatever it wants, including calling more *pcc* continuations (passing along the cc).