U4TGNN19D : hi all -- I'm using tools.cli and it's not clear how to pass multiple arguments to an option -- eg if I want an option to become a vector of numbers
U4TGNN19D : nevermind -- ended up taking it as a string and doing this - `:parse-fn (fn [input ](into [] (map #(Integer/parseInt (str %)) input)))]`
U2TCUSM2R : This is rather odd, but has anyone experienced an issue where dereferencing agents of collections truncates some of the elements?
U051SS2EU : remember that send and send-off can return before their action finishes - or even before it starts sometimes
U051SS2EU : ```+user=> (let [a (agent []) f #(send a conj nil) g (fn [] (count @a))] (dotimes [_ 10000] (f)) (repeatedly 10 g))(6622 6687 6716 6735 6757 6776 6794 6825 6851 6880)
```

U2TCUSM2R : <@U051SS2EU> that's got to be it. I think I just got confused because I'm creating and returning the agent from inside a macro. That's where I see the trouble. If I create a `def` for the return value and then `deref` that it waits just long enough for the last results
U051SS2EU : there's the function `await`
U2TCUSM2R : Thanks, Justin! :slightly_smiling_face:
U2TCUSM2R : Oh...actually I didn't realize `await` depends on the `:done` key. I'm using an `int-map` so no keywords...
U051SS2EU : what?
U051SS2EU : there's no magic :done key, await just lines up a lock that won't unlock until every pending action is done, then waits on it ```(defn await  "Blocks the current thread (indefinitely!) until all actions
  dispatched thus far, from this thread or agent, to the agent(s) have
  occurred.  Will block on failed agents.  Will never return if
  a failed agent is restarted with :clear-actions true."
  {:added "1.0"
   :static true}
  [&amp;amp; agents]
  (io! "await in transaction"
    (when *agent*
      (throw (new Exception "Can't await in agent action")))
    (let [latch (new java.util.concurrent.CountDownLatch (count agents))
          count-down (fn [agent] (. latch (countDown)) agent)]
      (doseq [agent agents]
        (send agent count-down))
      (. latch (await)))))
```

U2TCUSM2R : Oh, ok. I was confused by Clojure Docs: <https://clojuredocs.org/clojure.core/await>
U2TCUSM2R : I'll have to see why it's not still working
U051SS2EU : one possibility is that something is calling send after you call await?
U2TCUSM2R : I would assume
U2TCUSM2R : Here's the macro:
U2TCUSM2R : I think `await` isn't doing anything since it can't know whether `pdiff` has finished
U051SS2EU : it does do something - it waits on all actions that have been submitted before it is called - but you need something different if you want to wait on things pdiff might initiate after the point await is called
U051SS2EU : also calling def on the output of gensym is super weird -why not a let block?
U2TCUSM2R : I'm confused as to what you mean in your second statement
U051SS2EU : def always creates namespace level bindings, using def with a gensym is weird because how would you even know which thing to look for later?
U051SS2EU : let is for locals, so I would expect let instead in a scenario like this
U2TCUSM2R : Oh, the gensym isn't quasiquoted
U051SS2EU : that has nothing to do with it
U051SS2EU : def creates a new var in your namespace
U051SS2EU : oh - I see, you create the gensym def once
U051SS2EU : that guarantees that pdiff-once is a race condition if it's used in two threads though
U0BKWMG5B : What's the purpose of the macro?
U051SS2EU : that's another good question
U2TCUSM2R : The purpose of the macro is to automate three calls in one: creating the agent, calling pdiff, and returning the contents of the agent
U051SS2EU : if two threads use that macro, the second one will replace the data used by the first

U051SS2EU : it's extremely unsafe
U0BKWMG5B : Okay, but why not:```
(defn pdiff-once [poly order]
  (let [tape (agent (i/int-map))]
    (pdiff poly tape order)
    (await tape)
    @tape))
```

U0CGFT70T : why this: ```router=> (if-let [{params :params} {}] (str "params:" params ".") "NOT-defined")
"params:."```

U0BKWMG5B : <@U0CGFT70T> because {} is not falsey.
U2TCUSM2R : weavejester: let me try that. I'm not sure it'll work, but I'm relatively new to agents.
U0BKWMG5B : `(if-let [params (:params {})] ...)`
U0CGFT70T : <@U0BKWMG5B> : thanks :slightly_smiling_face:
U0BKWMG5B : With if-let, the expression on the right needs to be `nil` or `false` to fail
U0CGFT70T : trying to hit else if params not defined
U0CGFT70T : <@U0BKWMG5B> ok...
U0BKWMG5B : <@U2TCUSM2R> You could also use a promise.
```