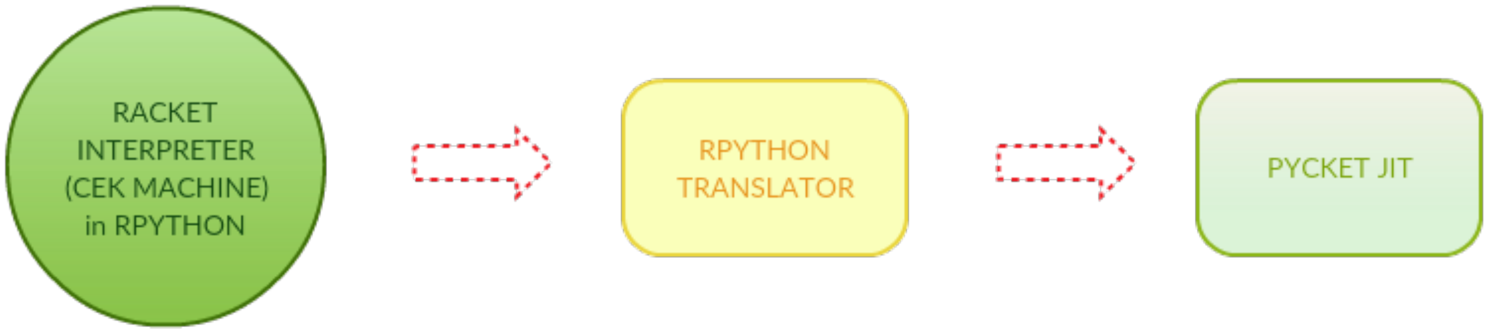
Cheating on Racket : One hack lead to another..

Caner Derici

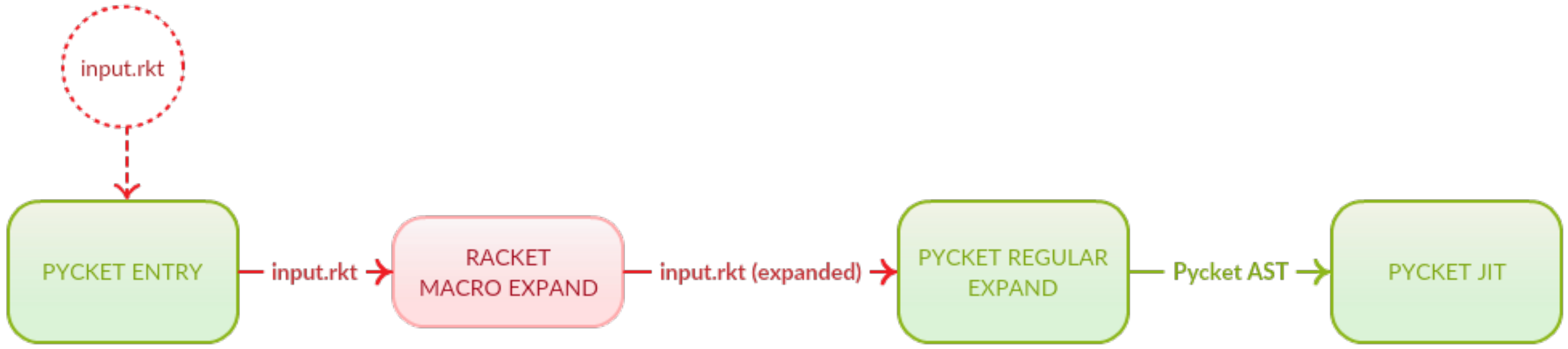
Nov 2017 @ wonks

Pycket

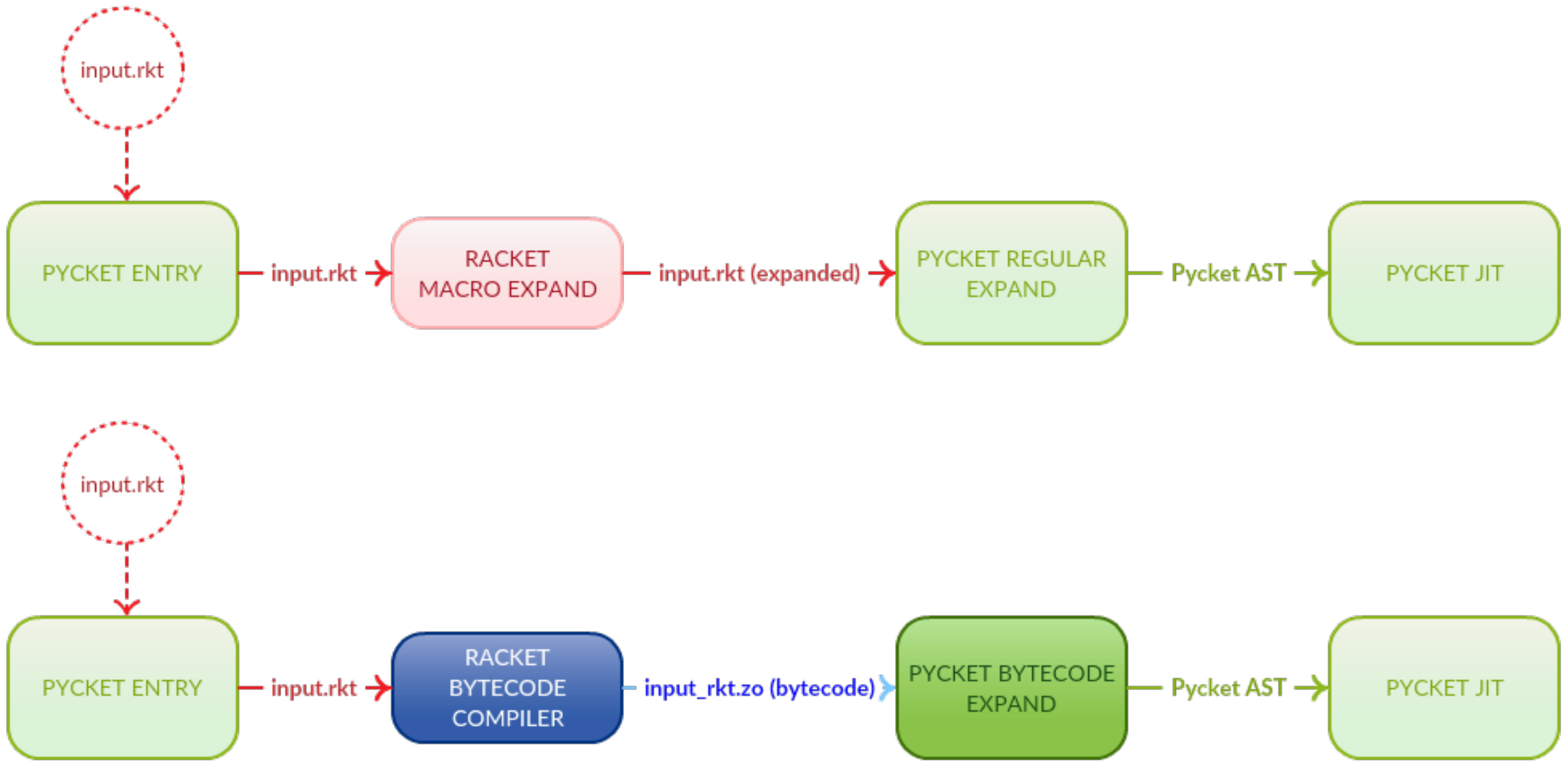
A JIT compiler generated by RPython framework.



Pycket AST Expansion : The Old Way



Pycket AST Expansion : The Old Way

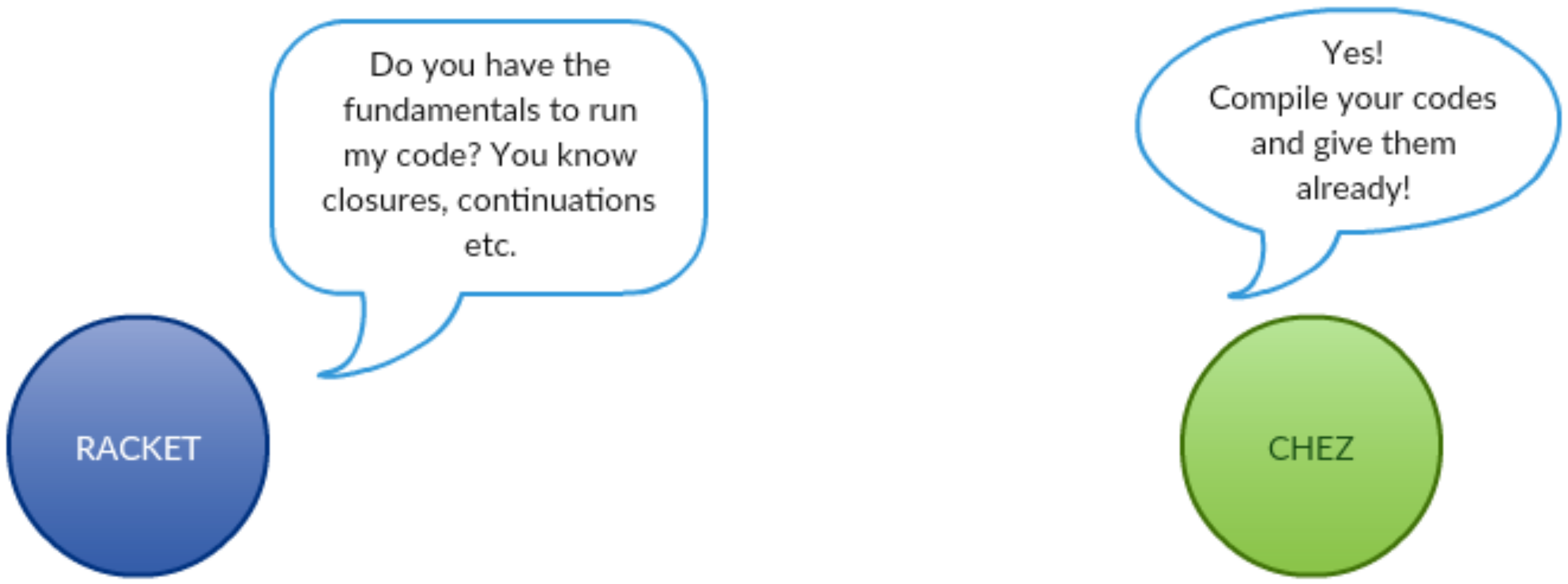


Racket Decides to Marry Chez

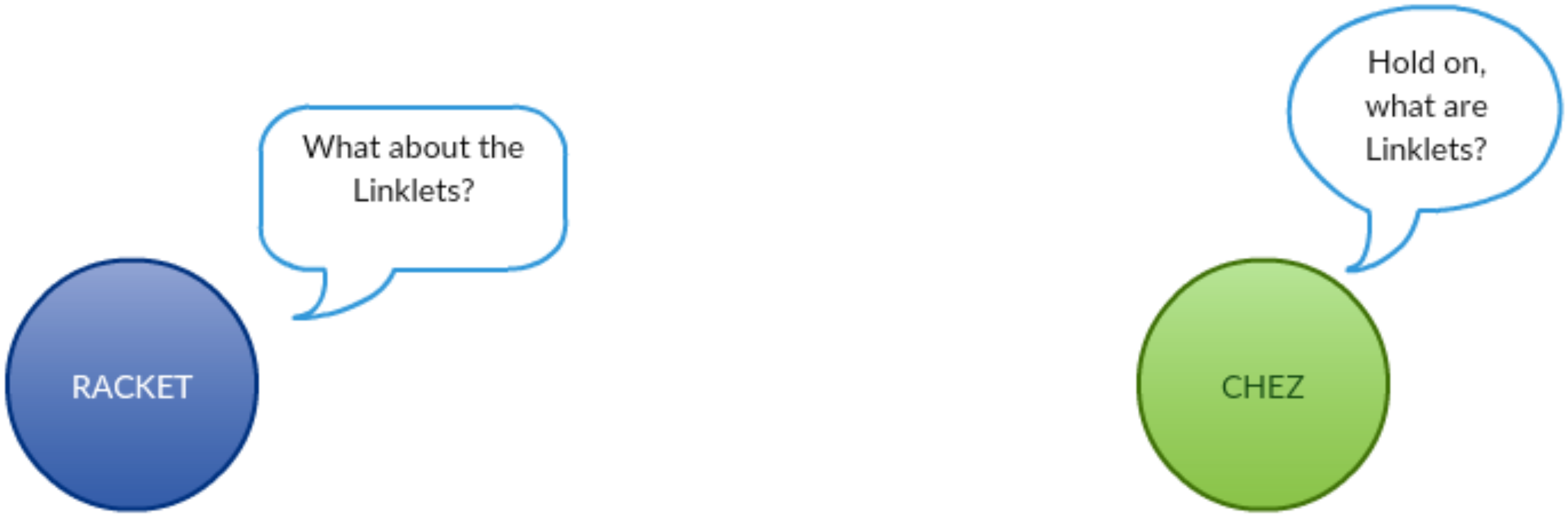
Dump the old generic JIT runtime.

Why?

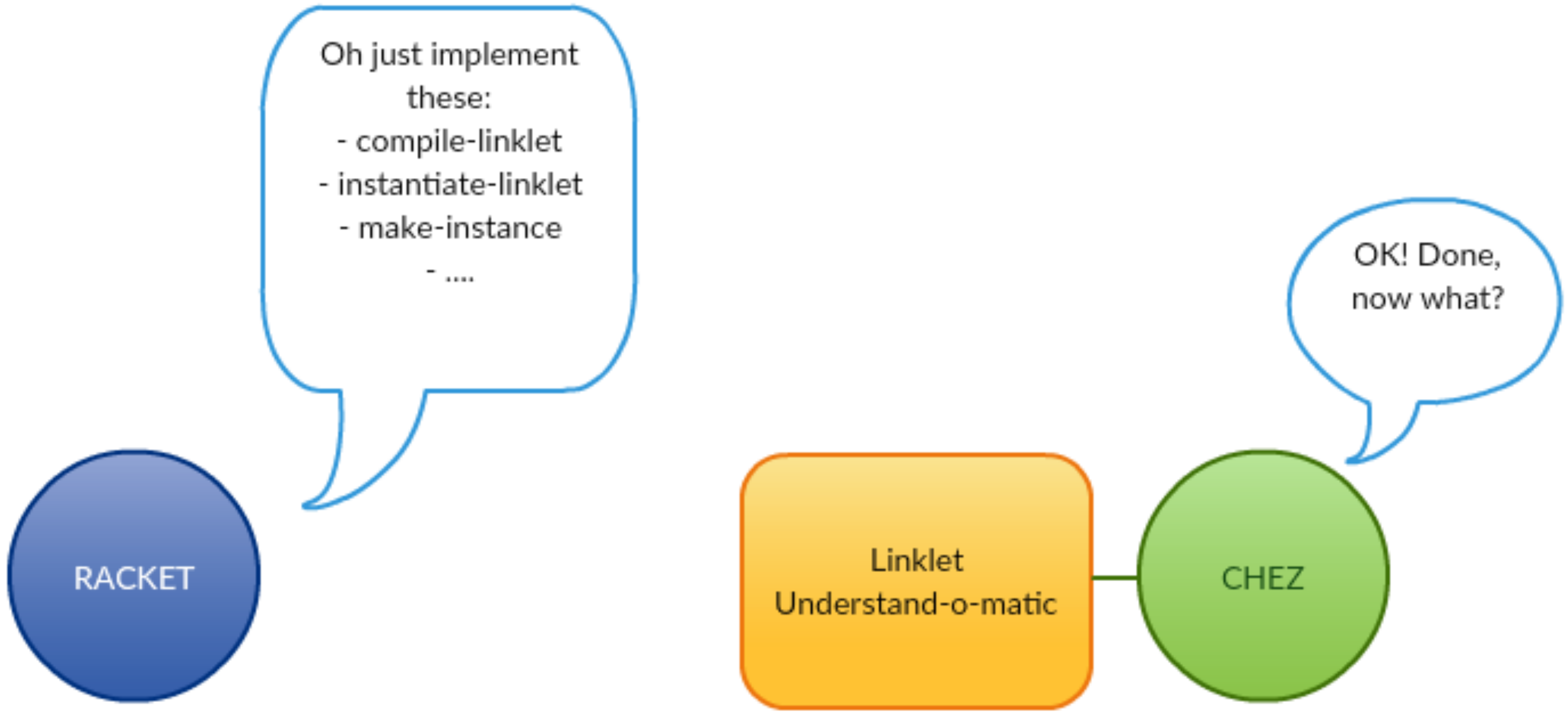
Finding The Middle Ground


Do you have the fundamentals to run my code? You know closures, continuations etc.
Yes! Compile your codes and give them already!
RACKET
CHEZ

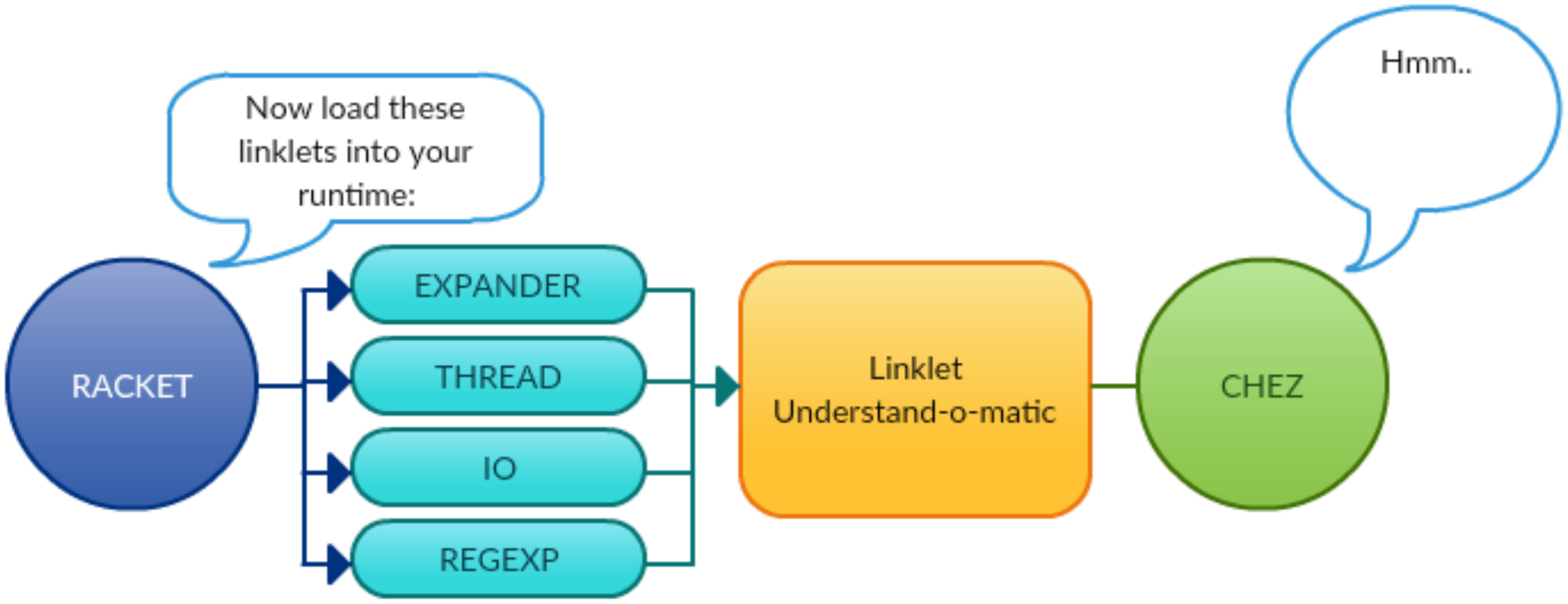
Finding The Middle Ground



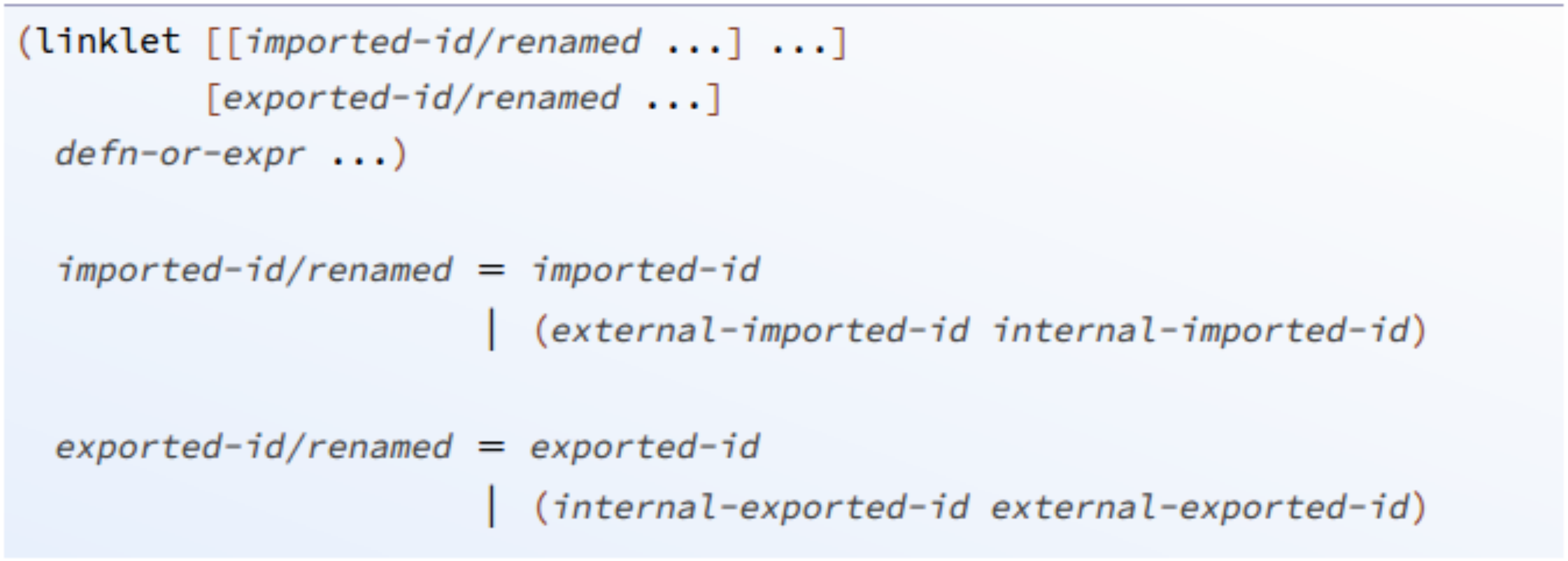
Finding The Middle Ground



Finding The Middle Ground


Now load these linklets into your runtime:
Hmm..
RACKET
EXPANDER
THREAD
IO
REGEXP
Linklet Understand-o-matic
CHEZ

Linklets

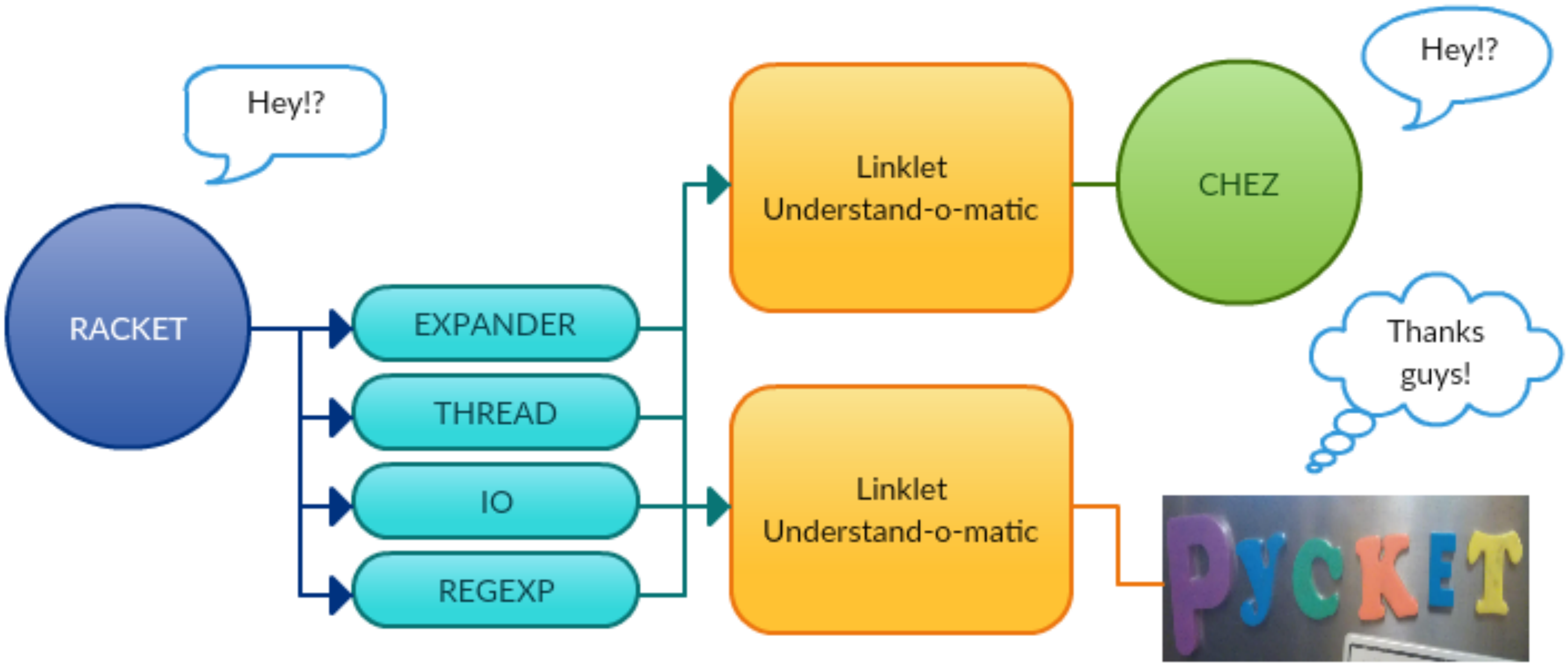


```
(linklet [[imported-id/renamed ...] ...]
         [exported-id/renamed ...]
  defn-or-expr ...)

  imported-id/renamed = imported-id
                      | (external-imported-id internal-imported-id)

  exported-id/renamed = exported-id
                      | (internal-exported-id external-exported-id)
```

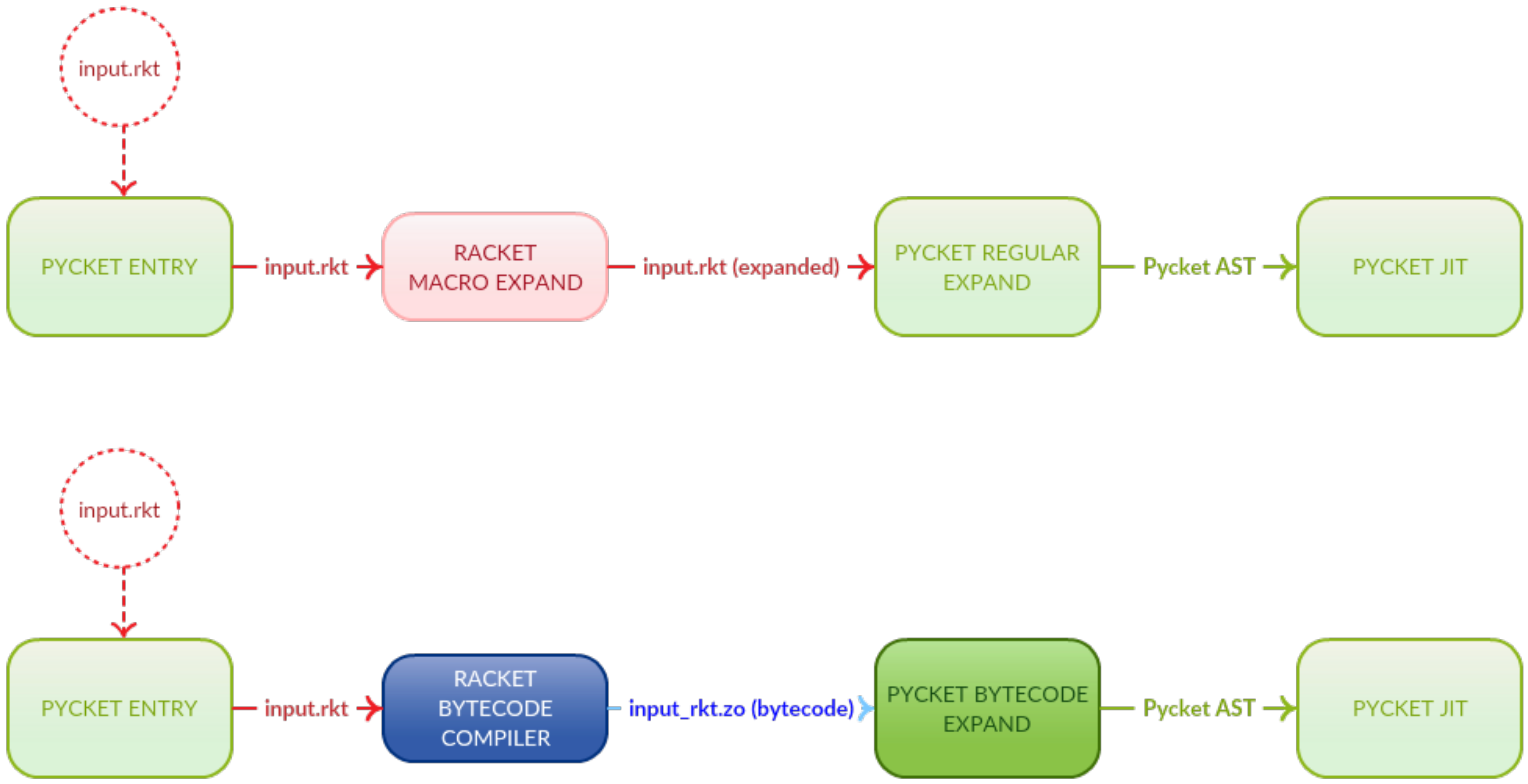
Expander

```
(linklet
 ()
 ((1/module-path-index? module-path-index?)
  (1/identifier-binding identifier-binding)
  (1/boot boot)
  (1/dynamic-require dynamic-require)
  (1/namespace-require namespace-require)
  (1/read read)
  (1/read-syntax read-syntax)
  (expand$1 expand)
  (1/eval eval)
  ...)
 (define-values
   (expand$1)
   (let-values (((expand40_0)
                 (lambda (s39_0 ns31_2 log-expand?32_0 to-parsed?33_0
                          serializable?34_0 ns35_0 log-expand?36_0
                          to-parsed?37_0 serializable?38_0)
                   ...)))))
 (define-values
   (1/eval)
   (let-values (((eval6_0)
                 (lambda (s5_0 ns1_4 compile2_0 ns3_0 compile4_0)
                   (let-values (((s_142) s5_0)) ...))))))
 ...)
```

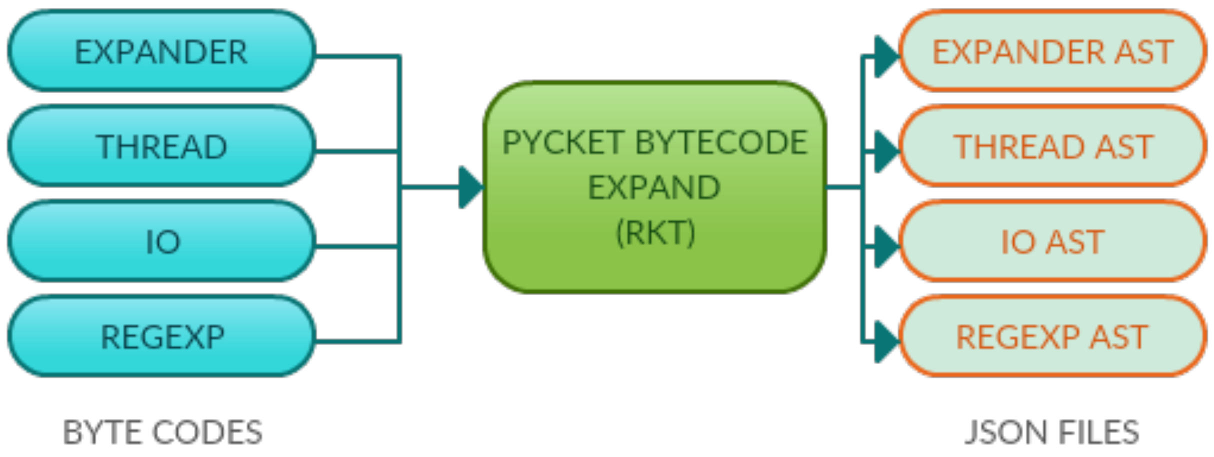
Pycket Understands Linklets Too!

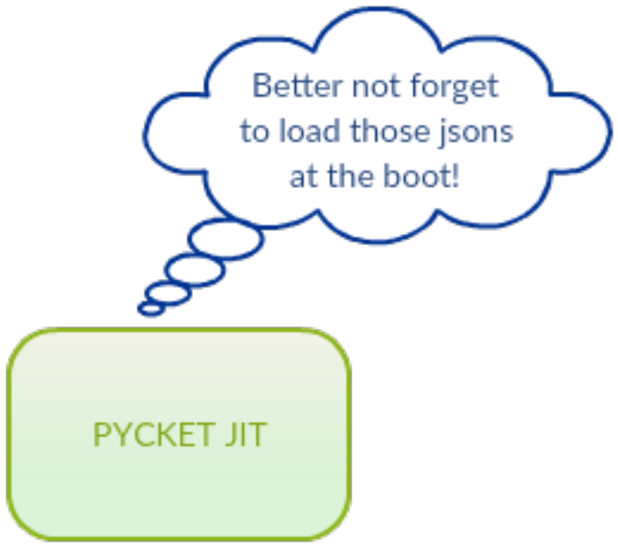


Pycket AST Expansion : The Old Way



Let’s Call Racket’s Expand!





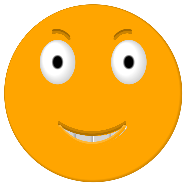
Let's Call Racket's Expand!

```
(define-values
  (expand$1)
  (let-values (((expand40₀)
                (lambda (s39₀ ns31₂ log-expand?32₀ to-parsed?33₀ ...)
                  <expand-body-rkt>)))
    <let-body-rkt>))
```

Racket

```
(define-values-name expand$1)
(define-values-body
  [(let-bindings
    ([expand40₀ (lambda (s39₀ ns31₂ log-expand?32₀ to-parsed?33₀ ...)
                  <expand-body-pycket)])
    (let-body
     <let-body-pycket>))])
```

Pycket



Let’s Call Racket’s Expand! : Ready

```
(linklet
 ()
 ((1/module-path-index? module-path-index?)
  (1/identifier-binding identifier-binding)
  (1/boot boot)
  (1/dynamic-require dynamic-require)
  (1/namespace-require namespace-require)
  (1/read read)
  (1/read-syntax read-syntax)
  (1/expand expand)
  (1/eval eval)
  ...)
       ...)
```

```
(define (boot)
  (seal)
  (current-module-name-resolver standard-module-name-resolver)
  (current-load/use-compiled default-load/use-compiled)
  (current-reader-guard default-reader-guard)
  (current-eval default-eval-handler)
  (current-compile default-compile-handler)
  (current-load default-load-handler)
  (current-read-interaction default-read-interaction))
```

Let’s Call Racket’s Expand! : Set

```
(linklet
 ()
 ((1/module-path-index? module-path-index?)
  (1/identifier-binding identifier-binding)
  (1/boot boot)
  (1/dynamic-require dynamic-require)
  (1/namespace-require namespace-require)
  (1/read read)
  (1/read-syntax read-syntax)
  (1/expand expand)
  (1/eval eval)
  ...)
       ...)
```

```
(namespace-require ''#%kernel)
```

Let’s Call Racket’s Expand! : Go

```
(linklet
 ()
 ((1/module-path-index? module-path-index?)
  (1/identifier-binding identifier-binding)
  (1/boot boot)
  (1/dynamic-require dynamic-require)
  (1/namespace-require namespace-require)
  (1/read read)
  (1/read-syntax read-syntax)
  (1/expand expand)
  (1/eval eval)
  ...)
       ...)
```

```
(eval
 (expand
  (read
   (open-input-string "(expt 2 3)"))))
```

We could also...

```
(linklet
 ()
 ((1/module-path-index? module-path-index?)
  (1/identifier-binding identifier-binding)
  (1/boot boot)
  (1/dynamic-require dynamic-require)
  (1/namespace-require namespace-require)
  (1/read read)
  (1/read-syntax read-syntax)
  (expand$1 expand)
  (1/eval eval)
  ...)
       ...)
```

```
(dynamic-require 'racket/base 'read-eval-print-loop)
```

Thanks!

Caner Derici

cadr/cderici

https://github.com/pycket/pycket/

IRC: freenode #pycket

Slack: racket #linklet

Inside Racket Seminars :

https://youtube.com/user/racketlang