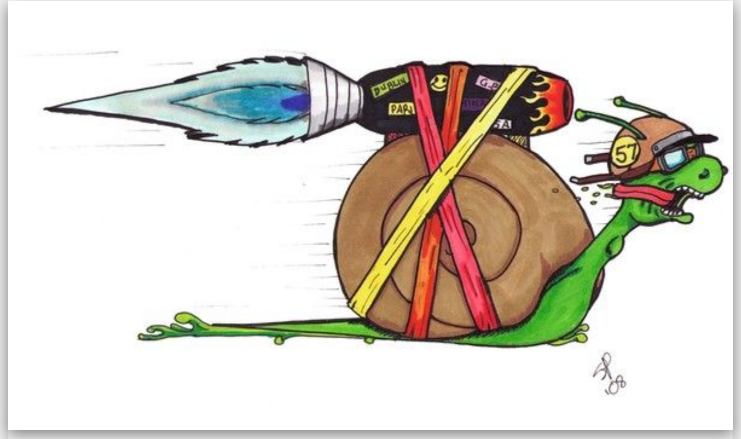
MeTTa-morph

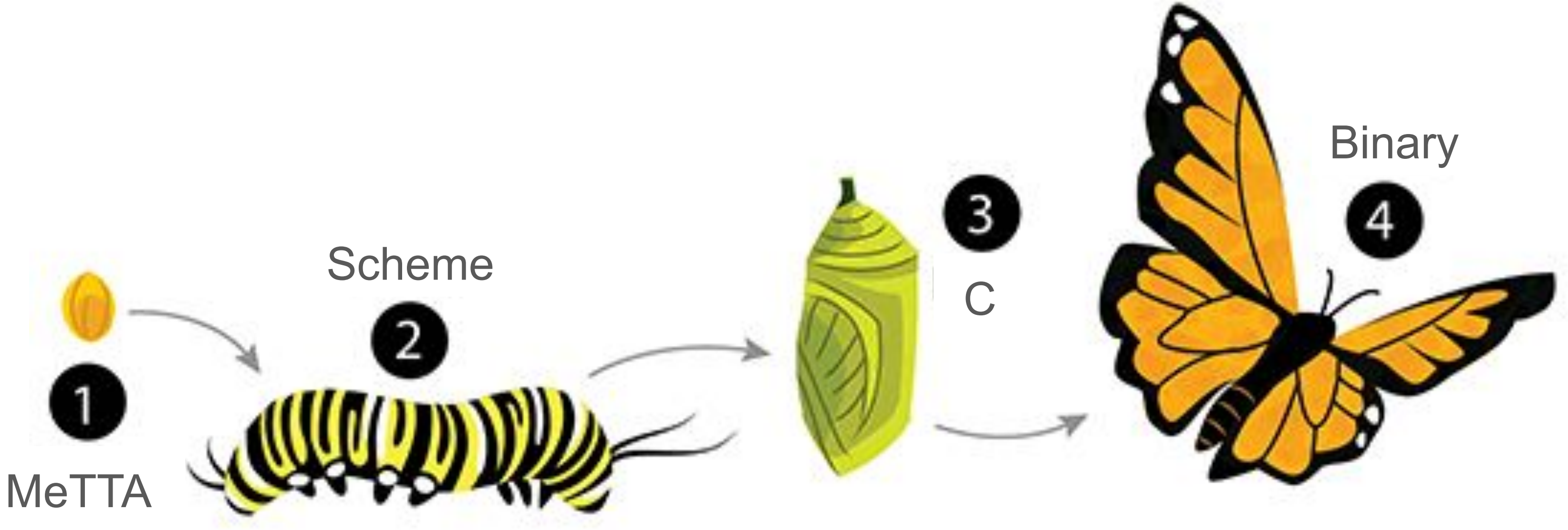
Presenters: Patrick Hammer, Peter Isaev

Design goals

- Speed up a MeTTa subset with a compiler
- Not a full MeTTa replacement!
- Allow conveniently calling compiled MeTTa functions from MeTTa interpreter



Multi-step compilation



1→2: Transformation with Scheme Macros
2→3: Chicken Scheme compiler
3→4: GCC/Clang C compiler

Hyperon integration

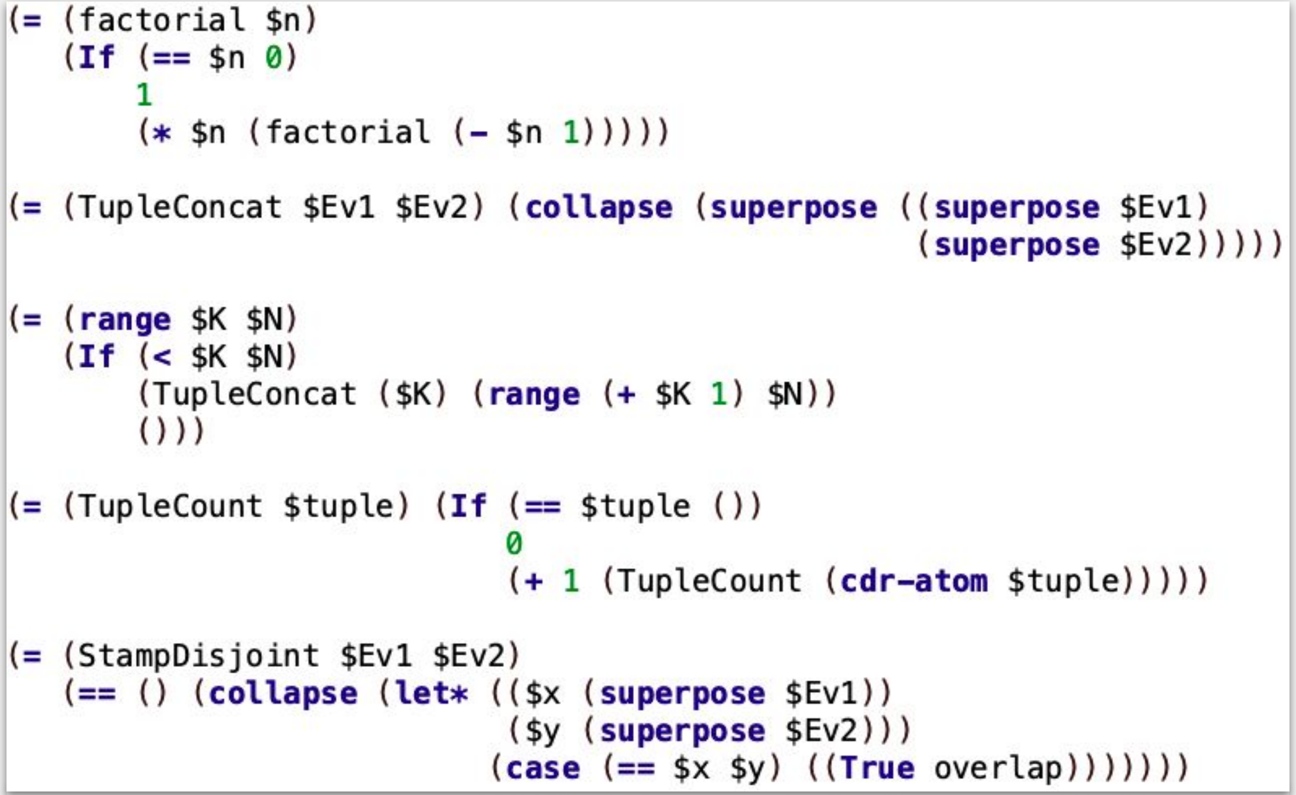
- Easily integrated with Hyperon API: **!(extend-py! mettamorph)**

- Allows speeding up crucial components considerably in the short-term

- Compilation invoked via **!(compile “code”)**, no plumbing needed!

Timing

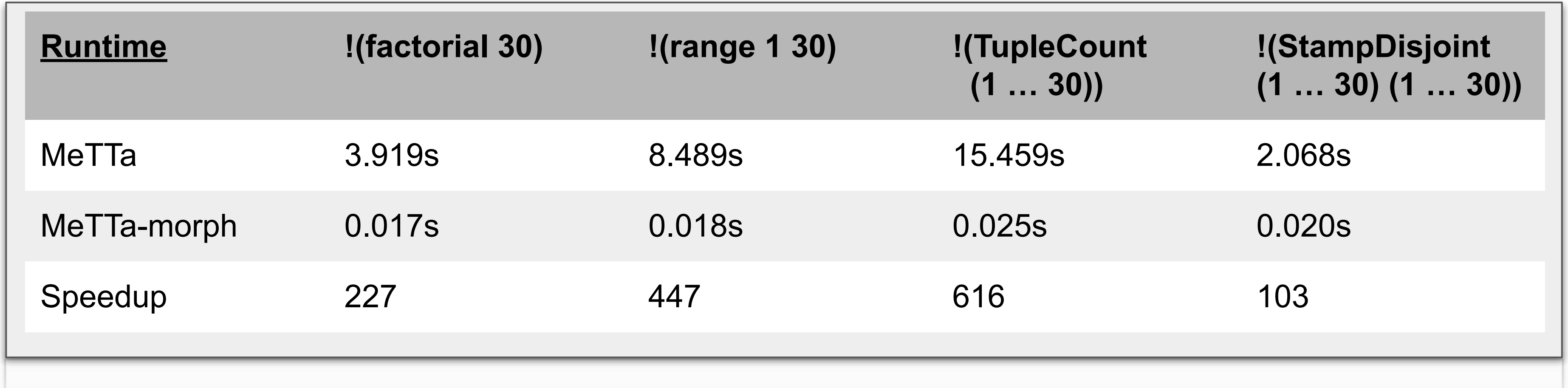
- ➢ Tail-recursive factorial and tuple element counting
- ➢ Range via tuple concatenation
- ➢ Non-recursive tuple disjoint check



```
(= (factorial $n)
   (If (== $n 0)
       1
       (* $n (factorial (- $n 1)))))

(= (TupleConcat $Ev1 $Ev2) (collapse (superpose ((superpose $Ev1)
                                                 (superpose $Ev2)))))

(= (range $K $N)
   (If (< $K $N)
       (TupleConcat ($K) (range (+ $K 1) $N))
       ()))

(= (TupleCount $tuple) (If (== $tuple ())
                           0
                           (+ 1 (TupleCount (cdr-atom $tuple)))))

(= (StampDisjoint $Ev1 $Ev2)
   (== () (collapse (let* (($x (superpose $Ev1))
                           ($y (superpose $Ev2)))
                          (case (== $x $y) ((True overlap)))))))
```

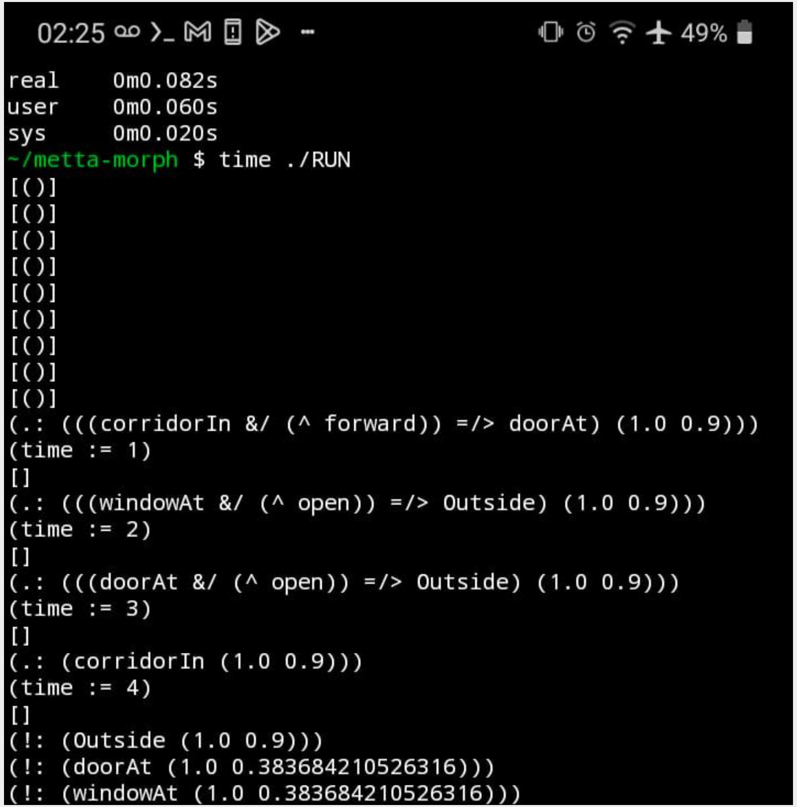
Speedup

- ➢ MeTTa-morph extension is ready to be used!
- ➢ Speedup factor usually in the range of hundreds!

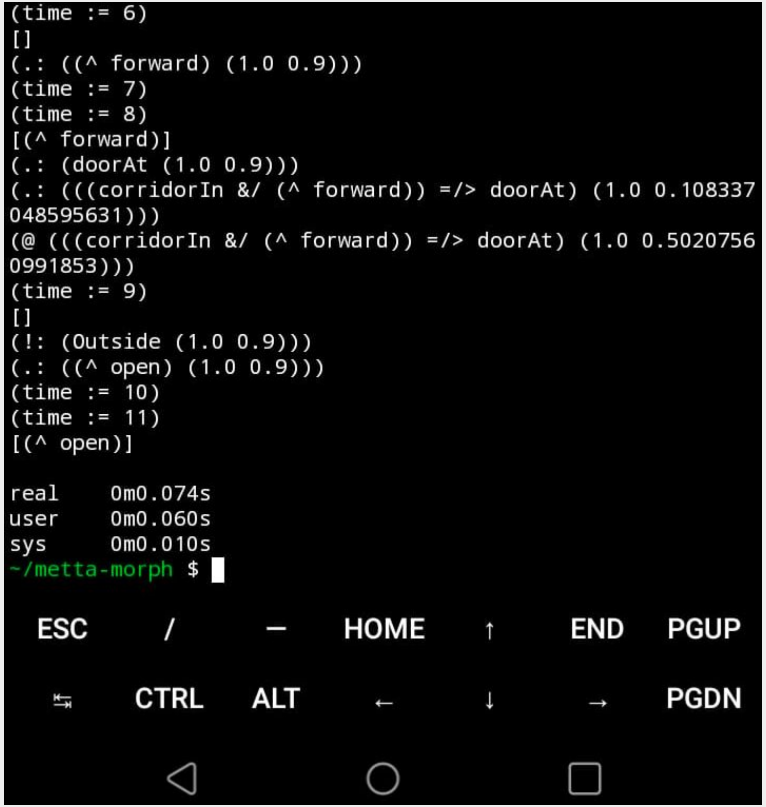


Runtime	**!(factorial 30)**	**!(range 1 30)**	**!(TupleCount (1 … 30))**	**!(StampDisjoint (1 … 30) (1 … 30))**
MeTTa	3.919s	8.489s	15.459s	2.068s
MeTTa-morph	0.017s	0.018s	0.025s	0.020s
Speedup	227	447	616	103

But can it run MeTTa-NARS? YES!



```
real    0m0.082s
user    0m0.060s
sys     0m0.020s
~/metta-morph $ time ./RUN
[()]
[()]
[()]
[()]
[()]
[()]
[()]
[()]
[()]
(.: (((corridorIn &/ (^ forward)) =/> doorAt) (1.0 0.9)))
(time := 1)
[]
(.: (((windowAt &/ (^ open)) =/> Outside) (1.0 0.9)))
(time := 2)
[]
(.: (((doorAt &/ (^ open)) =/> Outside) (1.0 0.9)))
(time := 3)
[]
(.: (corridorIn (1.0 0.9)))
(time := 4)
[]
(!: (Outside (1.0 0.9)))
(!: (doorAt (1.0 0.383684210526316)))
(!: (windowAt (1.0 0.383684210526316)))
```



```
(time := 6)
[]
(.: ((^ forward) (1.0 0.9)))
(time := 7)
(time := 8)
[(^ forward)]
(.: (doorAt (1.0 0.9)))
(.: (((corridorIn &/ (^ forward)) =/> doorAt) (1.0 0.108337
048595631)))
(@ (((corridorIn &/ (^ forward)) =/> doorAt) (1.0 0.5020756
0991853)))
(time := 9)
[]
(!: (Outside (1.0 0.9)))
(.: ((^ open) (1.0 0.9)))
(time := 10)
(time := 11)
[(^ open)]

real    0m0.074s
user    0m0.060s
sys     0m0.010s
~/metta-morph $
```

Takeaways

- Fast execution of MeTTa is possible and demonstrated!
- Potential design aspects could be shared with the Rholang translation effort?
- Limitations are listed in the repository.

Repository: https://github.com/patham9/metta-morph

Thank you!

Presenters: Patrick Hammer, Peter Isaev