The MeTTa calculus

L.G. Meredith[1]

CEO, F1R3FLY.io 9336 California Ave SW, Seattle, WA 98103, USA,
ceo@f1r3fly.io

Abstract. We describe a core calculus that captures the operational semantics of the language MeTTa.

1 Introduction and motivation

In [1] we described a register machine based operational semantics for MeTTa. While this has some utility for proving the correctness of compilers, it is not conducive for reasoning about types and other more abstract aspects of MeTTa-based computation. Here we present a calculus, together with a efficient implementation and prove the correctness of the implementation.

Additionally, we use the OSLF algorithm developed by Meredith, Stay, and Williams to calculate a spatial-behavioral type system for MeTTa. Further, we adapt a well known procedure for proving the termination of rewrites to provide a token-based security model for the calculus and implementation.

2 A symmetric reflective higher order concurrent calculus with backchaining

PROCESS
$$P, Q ::= 0 \mid \mathsf{for}(t \leftrightarrow x)P \mid x?P \mid *x \mid P|Q$$

NAME
$$x, y ::= @P$$

TERM
$$t, u ::= atom \mid (t^*)$$

TERM
$$atom ::= x \mid \mathsf{Bool} \mid \mathsf{String} \mid \mathsf{Int} \mid P$$

EQUIV
$$P|0 \equiv P \quad P|Q \equiv Q|P \quad P|(Q|R) \equiv (P|Q)R$$

ALPHA
$$\frac{\mathsf{occurs}(t, y)}{\mathsf{for}(t \leftrightarrow x)P \equiv \mathsf{for}(t\{z/y\} \leftrightarrow x)(P\{z/y\}) \text{ if } z \notin \mathsf{FN}(P)}$$

$$\text{COMM}\quad \frac{\sigma = \mathsf{unify}(t, u)}{\mathsf{for}(t\leftarrow\!\!\rightarrow x)P \mid \mathsf{for}(u\leftarrow\!\!\rightarrow x)Q \to P\dot{\sigma}|Q\dot{\sigma}}$$

$$\text{PAR}\quad \frac{P \to P'}{P|Q \to P'|Q} \qquad \text{EQUIV}\quad \frac{P \equiv P' \qquad P' \to Q' \qquad Q' \equiv Q}{P \to Q}$$

$$\text{REFL}\quad \frac{P \to P'}{x?P \to x!(P')}$$

where $\dot{\sigma}$ denotes the substitution that replaces all variable to process bindings with variable to name bindings. Thus, $\{P/x\} = \{@P/x\}$.

2.1 Intuitive mapping to MeTTa

MeTTa language	MeTTa calculus
(addAtom *space term*)	$\mathsf{for}(term \leftarrow\!\!\rightarrow space)\mathsf{0}$
(remAtom *space term*)	$\mathsf{for}(term \leftarrow\!\!\rightarrow space)\mathsf{0}$
(? *space term*)	$\mathsf{for}(term \leftarrow\!\!\rightarrow space)\mathsf{0}$

3 Some useful features

3.1 Replication and freshness

$$\text{PROCESS}\quad P, Q ::= \ldots \mid\ !P \mid\ \mathsf{new}\ x\ \mathsf{in}\ \{\ P\ \}$$

In the core calculus, when two terms rendezvous at a space ($\mathsf{for}(t\leftarrow\!\!\rightarrow x)P \mid \mathsf{for}(u\leftarrow\!\!\rightarrow x)Q$) they are *consumed* and replaced by their continuations ($P\dot{\sigma}|Q\dot{\sigma}$). It is frequently useful in programming applications to leave one or the other in place. Thus, when $!\mathsf{for}(t\leftarrow\!\!\rightarrow x)P$ rendezvous with $\mathsf{for}(u\leftarrow\!\!\rightarrow x)Q$ it reduces to $!\mathsf{for}(t\leftarrow\!\!\rightarrow x)P|P\dot{\sigma}|Q\dot{\sigma}$.

Likewise, in programming applications it is often useful to guarantee that computations rendezvous in a private space. The state denoted by $\mathsf{new}\ x\ \mathsf{in}\ \{\ P\ \}$ guarantees that x is private in the scope P. Therefore, $\mathsf{new}\ x\ \mathsf{in}\ \{\ \mathsf{for}(t\leftarrow\!\!\rightarrow x)P \mid \mathsf{for}(u\leftarrow\!\!\rightarrow x)Q\}$ guarantees that the rendezvous happens in a private space.

3.2 Fork-join concurrency

This next bit of syntactic sugar illustrates the value of the for-comprehension. Specifically, it facilitates the introduction of fork-join concurrency, which is predominant in human

decision-making processes. The following syntax should be read as an expansion of the core calculus, *replacing* the much simpler for-comprehension with a more articulated one.

PROCESS
$$P, Q ::= \ldots \mid \mathsf{for}([\mathsf{Join}])P$$

JOINS
$$[\mathsf{Join}] ::= \mathsf{Join} \mid \mathsf{Join};[\mathsf{Join}]$$

JOIN
$$\mathsf{Join} ::= [\mathsf{Query}]$$

QUERIES
$$[\mathsf{Query}] ::= \mathsf{Query} \mid \mathsf{Query}\&[\mathsf{Query}]$$

QUERY
$$\mathsf{Query} ::= t \leftrightarrow x$$

In case the BNF is a little opaque, here is the template.

```
for(
  y11 <-> x11 & ... & ym1 <-> xm1 ; // received in any order,
     ... ; // but all received before the next row
  y1n <-> x1n & ... & ymn <-> xmn
){ P }
```

4 From calculus to efficient implementation

TBD

5 Tokenized security

TBD

6 Spatial-behavioral types

TBD

7 Stochastic and quantum execution

7.1 Stochastic execution

TBD

7.2 Quantum execution

TBD

8 Conclusions and future research

TBD

Acknowledgments. The author wishes to thank Mike Stay for his long time collaboration.

References

1. Adam Vandervorst Lucius Gregory Meredith, Ben Goertzel. Meta-metta: An operational semantics for metta, 2023.