Meta-Reasoning in MeTTa for Inference Control via Provably Pruning the Search Tree

Nil Geisweiller

SingularityNET Foundation,
Zug, Switzerland
nil@singularitynet.io

Abstract

The paper describes a meta-reasoning experiment to attempt to explore the possibility of improving reasoning efficiency via reasoning. A backward chainer implementation is provided in MeTTa, a novel programming language for AI. The backward chainer is modified to be user controllable. The backward chainer itself (though using a dedicated control theory) is then plugged into that controlling mechanism. A toy example is described. Then the paper concludes with related and future work.

1 Methodology

How to speed-up reasoning by reasoning about reasoning? This is the question we will attempt to begin to answer in the context of a meta-reasoning experiment described below.

1. A backward chainer is implemented in MeTTa [1] that takes in input a *problem* theory (a set of rewriting rules), a query (a typing relationship containing holes) and outputs solutions in the form of the query with its holes filled. By leveraging non-determinism and unification in MeTTa, a compact implementation is obtained, as evidenced by the code shown in Appendix A.

2. The backward chainer is then modified by inserting conditionals at every non-deterministic intersection, alongside user programmable contexts. These conditionals are in charge of deciding whether to prune or to continue a particular path of the backward chainer. The modified base case of the backward chainer is provided in Appendix B and the modified recursive step can be found under the Controlled Backward Chainer Section of [3].

3. *Continuation* predicates associated to these conditionals are themselves defined as backward chainer calls over a *control* theory, alongside a query formulating the question of whether to prune or to continue a particular branch. Thus, at every step during problem solving, the backward chainer over the control theory is called. If a proof of continuation is found, then the backward chainer over the problem theory is allowed to pursue its particular path, otherwise the branch is pruned.

2 Results

The experiment is carried over a toy problem, the relationship of precedence between the months of the year with a shortcut for the month of January. The control theory is simple and tailored toward that particular problem only, by formalizing the January shortcut. In that experiment the simplicity of the control theory allows to obtain proofs of continuation at a small cost. For that reason, the performances with inference control are dramatically improved, ranging from no speed-up to many-fold speed-ups depending on the difficulty of the query. More information about the experiment alongside its code can be found in [3].

3 Related Work

On the theoretical side, one ought to mention the work of Jürgen Schmidhuber on the Gödel Machine [7], a self-improving machine powered at its center by an Automated Theorem Prover (ATP) used to infer theorems expressing that a particular rewrite may lead to increasing the expected reward. In that paper, Jürgen mentions the possibility of using such machine to improve theorem proving itself. Worth mentioning is a survey of John Harrison on metatheory and reflection in theorem proving [6]. The survey is mostly theoretical and more focused on correctness of meta-reasoning than efficiency of inference. Some practical ideas are mentioned nonetheless such as inferring tactics that are correct by construction and using formalism with explicit reflection.

On the practical side, finding work on meta-reasoning in the context of ATP proved to be difficult. A recent survey [2] on applying AI techniques for guiding ATP, citing over 150 papers, did not seem to mention any work on meta-reasoning. Additional Internet search did not seem to yield more results. We may of course have missed relevant work due to not using the right terminology while searching. Regardless, it indicates that meta-reasoning in the context of ATP is likely under-explored and deserves more attention than currently given.

4 Conclusion and Future Work

The work described in this document is only barely scratching the surface. Below are a few suggestions for future research.

1. The control theory should capture the notion of inference control more broadly as opposed to being hardwired to a specific problem. This would allow to reuse the same control theory across multiple problems.

2. The control theory should be amenable to reason about empirical data. In this manner, traces left by the backward chainer while attempting to solve problems could be used to infer predictive patterns about the likelihood of success of pursuing particular paths. This could be done via inferential pattern mining [4], abstract reasoning, or combinations thereof by using an appropriate logic such as Probabilistic Logic Networks [5].

3. The control theory should be so broad and applicable that it may benefit inference control over the control theory itself, which would open the possibility for creating virtuous feedback loops of meta-reasoning.

References

[1] MeTTa Language Official Website. https://metta-lang.dev.

[2] Lasse Blaauwbroek, David Cerna, Thibault Gauthier, Jan Jakubruv, C. Kaliszyk, Martin Suda, and Josef Urban. Learning guided automated reasoning: A brief survey. *ArXiv*, abs/2403.04017, 2024.

[3] Nil Geisweiller. Inference Control Experiment in MeTTa. https://github.com/trueagi-io/chaining/tree/main/experimental/inference-control.

[4] Nil Geisweiller and Ben Goertzel. An inferential approach to mining surprising patterns in hypergraphs. In Patrick Hammer, Pulin Agrawal, Ben Goertzel, and Matthew Iklé, editors, *Artificial General Intelligence*, pages 59–69, Cham, 2019. Springer International Publishing.

[5] Ben Goertzel, Matthew Ikle, Izabela Freire Goertzel, and Ari Heljakka. *Probabilistic Logic Networks*. Springer US, 2009.

[6] John Harrison. Metatheory and reflection in theorem proving: A survey and critique. 1995.

[7] Jürgen Schmidhuber. Goedel machines: Self-referential universal problem solvers making provably optimal self-improvements. *ArXiv*, cs.LO/0309048, 2003.

A MeTTa Backward Chainer

The backward chainer, `bc`, takes three arguments, a knowledge base containing the theory `$kb`, a query containing a typing relationship between proof and theorem `(: $prf $thrm)`, and a maximum depth. It returns a non-deterministic superposition of results, the query with its variables substituted by MeTTa terms encoding wholes or parts of proofs and theorems. It has two cases

1. Base case: the query directly matches an element of the knowledge base.
2. Recursive step: the query is divided into two sub-queries, one for finding a proof abstraction and another one for finding a proof argument applied to that proof abstraction.

Unlike with traditional recursive algorithms, the base case and the recursive step are non-deterministically competing with each other. Also, in MeTTa the `let` operation corresponds to unification. The code is provided below and more information can be found in [3]

```
;; Base case
(= (bc $kb (: $prf $thrm) $_) (match $kb (: $prf $thrm) (: $prf $thrm)))
;; Recursive step
(= (bc $kb (: ($prfabs $prfarg) $thrm) (S $k))
   (let* (((: $prfabs (-> $prms $thrm)) (bc $kb (: $prfabs (-> $prms $thrm)) $k))
          ((: $prfarg $prms) (bc $kb (: $prfarg $prms) $k)))
     (: ($prfabs $prfarg) $thrm)))
```

B Base Case with Continuation Conditionals

The backward chainer is modified to take in inputs a *control structure*, a *context*, in addition to a control theory and a query. A continuation conditional is placed at the entry of the base case call, invoking `$bcont` in order to determine whether to continue or to prune the current branch. Another conditional is placed at the exit of the match query, invoking `$mcont` to determine whether to continue or to prune the current branch. It is important to have a conditional at the exit of the match query because at that point more information is available to the continuation predicate. The built-in function `empty` is used to prune the current branch. The modified base case can be found below and more information (including the modified recursive step) can be found in [3].

```
;; Modified base case
(= (bc $kb                                                  ; Knowledge base
       (MkControl $absupd $argupd $bcont $rcont $mcont)     ; Control
       $ctx                                                 ; Context
       (: $prf $thrm))                                      ; Query
   ;; Base case continuation conditional
   (if ($bcont (: $prf $thrm) $ctx)
       ;; Continue by querying the kb
       (match $kb (: $prf $thrm)
              ;; Match continuation conditional
              (if ($mcont (: $prf $thrm) $ctx)
                  ;; Continue by returning the queried result
                  (: $prf $thrm)
                  ;; Terminate by pruning
                  (empty)))
       ;; Terminate by pruning
       (empty)))
```