

Topos-Based Logical Frameworks and an Application to (Meta)Heuristic Search

Fernando Náufel do Amaral^{1,2}, Edward Hermann Haeusler¹

¹ TecMF, Depto. de Informática, PUC-Rio, Brazil

²Depto. de Informática e Ciência da Computação, IME, UERJ, Brazil

{fnaufel,hermann}@inf.puc-rio.br, fnaufel@ime.uerj.br

***Abstract.** We define some topoi related to the topos of forests and illustrate how the internal language of such topoi can be used (via local set theory) to define logics where we can represent (meta)heuristic search strategies for solving computational problems.*

1. Introduction

Higher-order languages have been frequently used as logical frameworks. From a logical point of view, one of the most interesting developments of topos theory [Goldblatt 1979] has been the investigation of the internal logics of topoi by means of local set theory (henceforth LST) [Bell 1988]. This is accomplished by viewing any topos as a model of a theory in the higher-order language of LST.

We develop a topos-based formal model for (meta)heuristics general enough to encompass the many (sometimes informal) definitions found in the literature. Among such definitions, we may quote, for example: “Heuristics are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal” [Pearl 1985]. According to [Voss 2001], a metaheuristic is “an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions.”

In a more abstract, unified view, both heuristics and metaheuristics are techniques for solving a given problem by first defining some kind of “space” inhabited by candidate answers (which are related by some structure imposed by the definition of such a space) and then defining a strategy for moving in the defined space in search of an appropriate answer. These two aspects — the search space and the search strategy — are the basis for our formal definition of (meta)heuristics.

The main advantage of using LST to construct a model of (meta)heuristics is the fact that, once we define an appropriate topos relating problems and search spaces, we immediately have at our disposal a logical theory whose model is the defined topos (and which can be represented in the internal language of the topos). This eliminates questions of expressiveness, consistency and completeness which would fall upon us if we were to construct our logical theory otherwise. In other words, the greatest benefit of LST is that the model (i.e., the defined topos) comes equipped with a logical language and theory of its own. In this sense, topoi and their internal languages function as logical frameworks.

One possible application of the present work lies in the logic-based development of software architectures for the implementation of (meta)heuristic search strategies. We

envison a framework (in the software engineering sense) where logical specifications of such strategies can be constructed and refined in order to generate executable code, possibly to be combined with predefined, domain-specific libraries. Thus, some heuristic concepts can be seen as images of the structure of the functor classifying them in the space of (possible and viable) heuristics. This is a step toward an architectural view of heuristics design.

2. Defining the Topos

We assume the reader is familiar with basic categorical and topos-theoretical notions such as objects, morphisms, (contravariant) functors, natural transformations etc. A comprehensive reference on the subject is [Goldblatt 1979].

Definition 2.1 (Problems). *A problem is a triple $P = \langle D, R, p \rangle$, with D and R countable, nonempty sets, and $p \subseteq D \times R$ a relation. Elements $d \in D$ are called instances; elements $r \in R$ are answers; $(d, r) \in p$ means that r is a correct answer for instance d .*

Definition 2.2 (Reductions). *Given $P = \langle D, R, p \rangle$ and $P' = \langle D', R', p' \rangle$, a reduction $P \xrightarrow{(\tau, \sigma)} P'$ consists of a pair (τ, σ) of functions computable in polynomial time, with $\tau : D \rightarrow D'$ and $\sigma : R' \rightarrow R$ such that correct answers are preserved; more precisely, for every $d \in D$ and every $r' \in R'$, we have that $(\tau(d), r') \in p' \Rightarrow (d, \sigma(r')) \in p$.*

Definition 2.3 (The category **Prob** of problems). ***Prob** is a designated thin, skeletal category having as objects the set of problems of interest and having as morphisms the set of reductions of interest.*

The search spaces we will define for the problems in **Prob** are based on forests. A convenient category-theoretical definition of forests and forest homomorphisms is the following:

Definition 2.4 (Forests, forest homomorphisms). *A forest S is a functor $S : \omega^{op} \rightarrow \mathbf{Set}$, where ω^{op} is the category $0 \leftarrow 1 \leftarrow 2 \leftarrow \dots$. A homomorphism h from a forest S to a forest S' is a natural transformation $h : S \rightarrow S'$.*

To see how a functor $S : \omega^{op} \rightarrow \mathbf{Set}$ defines a forest, one views $S(0)$ as the set of root vertices of the forest, $S(1)$ as the set of vertices on the second level of the forest, and so on. The functor S will map the morphism $0 \leftarrow 1$ to the function $parent_1 : S(1) \rightarrow S(0)$, the morphism $1 \leftarrow 2$ to the function $parent_2 : S(2) \rightarrow S(1)$ and so on.

Definition 2.5 (The category **Forest**). *The category **Forest** of forests and forest homomorphisms is the functor category $\mathbf{Set}^{\omega^{op}}$.*

A search space for a problem $P = \langle D, R, p \rangle$ will be an object FP of **Forest**, enriched with some additional information about P . More specifically, we want to label each vertex v of the forest FP with a set of answers of P (i.e., with a set $\lambda(v)$ such that $\lambda(v) \subseteq R$). This is done via a natural transformation to the contravariant functor defined below.

Definition 2.6 (The L functor). *$L : \mathbf{Prob}^{op} \rightarrow \mathbf{Forest}$ is the functor that maps a problem $P = \langle D, R, p \rangle$ to the forest LP with infinitely many levels, whose set of vertices at level 0 is $\wp(R)$ (the powerset of R), and whose set of vertices at level i , for $i > 0$, is $\wp(R)^{i+1}$, the set of all $(i + 1)$ -tuples whose components are sets of answers in R . As for morphisms,*

L maps a reduction $P \xrightarrow{(\tau, \sigma)} P'$ to the forest homomorphism $LP' \xrightarrow{L(\tau, \sigma)} LP$ such that a vertex (A'_1, \dots, A'_i) at the i th level of LP' is mapped to the vertex $(\sigma(A'_1), \dots, \sigma(A'_i))$ at the i th level of LP .

The labeling of vertices of FP with sets of answers of P is justified as follows: (a) in population-based metaheuristics, each vertex in the search space corresponds to a population, which is a set of answers; (b) in local search metaheuristics and transformation heuristics, each vertex corresponds to a single answer r , which can be seen as the singleton $\{r\}$; and (c) in constructive heuristics, each vertex corresponds to a “partial” answer, and moving from one vertex to another corresponds to adding “elements” to the partial answer in order to build a complete answer. It can be shown that a partial answer of this kind can be represented by a set A of answers, and the addition of “elements” to an answer can be seen as the selection of a proper subset of A , in a process sometimes called *pruning*.

Definition 2.7 (Answer forest assignment). *An answer forest assignment is a pair $\langle F, \lambda \rangle$, where $F : \mathbf{Prob}^{op} \rightarrow \mathbf{Forest}$ is a functor and $\lambda : F \rightarrow L$ is a natural transformation.*

Take a problem P in \mathbf{Prob} ; to see how λ_P corresponds to a labeling of the vertices of FP , note that a vertex v at the i th level of FP with $\lambda(v) = (A_1, \dots, A_i)$ is considered to be labeled by the set A_i . The other components A_1, \dots, A_{i-1} store the labels of the ancestor vertices of v from the root down to the parent of v . Furthermore, because λ is a natural transformation, the labeling of the forests assigned to different problems in \mathbf{Prob} is done consistently (i.e., naturally).

Proposition 2.1. *The collection of all answer forest assignments is the collection of objects of the slice category $(\mathbf{Forest}^{\mathbf{Prob}^{op}}) \downarrow L$, where $\mathbf{Forest}^{\mathbf{Prob}^{op}}$ is the category of all contravariant functors from \mathbf{Prob} to \mathbf{Forest} and L is the functor introduced in Def. 2.6.*

Theorem 2.1. *Let this category of answer forest assignments be called \mathbf{AFA} . The category \mathbf{AFA} is a topos.*

Proof (sketch). $\mathbf{Forest}^{\mathbf{Prob}^{op}}$ is isomorphic to the functor category $(\mathbf{Set}^{\omega^{op}})^{\mathbf{Prob}^{op}}$, which, in turn, is isomorphic to the functor category $\mathbf{Set}^{\omega^{op} \times \mathbf{Prob}^{op}}$, which is a presheaf topos. Therefore, \mathbf{AFA} is a slice category of a topos, and hence itself a topos (via the fundamental theorem of topoi — see [Bell 1988]). \square

Each object of \mathbf{AFA} corresponds, thus, to an assignment of search spaces to the problems in \mathbf{Prob} . These assignments are related via the morphisms of \mathbf{AFA} . Theorem 2.1 is important because it allows us to describe these assignments and relationships in the internal language of the topos \mathbf{AFA} , using local set theory (LST).

Actually, Theorem 2.1 gives us two topoi, namely, $\mathbf{Forest}^{\mathbf{Prob}^{op}}$, whose objects assign *unlabeled* forests to problems; and its slice topos \mathbf{AFA} , whose objects assign *labeled* forests to problems. As always happens in such cases, the internal logic of the slice topos \mathbf{AFA} can be represented in the internal logic of the unsliced topos $\mathbf{Forest}^{\mathbf{Prob}^{op}}$. For technical reasons not explained here, it is more convenient to work within the internal logic of the latter (unsliced) topos; so, in what follows, we will be using the internal logic of $\mathbf{Forest}^{\mathbf{Prob}^{op}}$ to manipulate objects and morphisms of \mathbf{AFA} in order to describe (meta)heuristic search strategies.

3. Representing Search Strategies

Any topos can be seen as a model of some local set theory (LST). In LST, the notion of *set* is replaced by that of *type*. In the language of LST each term (including those representing sets) has an associated type. The “local” in LST means that some common set-theoretical operations, such as union, intersection etc., are only defined for terms of the same type (i.e., locally). Apart from that, the language is very similar to that of set theory, with primitive symbols $=$, \in and $\{ \mid \}$. The language of LST is defined in [Bell 1988, p. 68ff]. There, it is also shown how such a language can be interpreted in an arbitrary topos.

As explained in Sec. 2, we presuppose a thin, skeletal category \mathbf{Prob} . Due to space limitations, we simplify our exposition here by considering $\mathbf{Prob} = \bullet$; i.e., the domain of all our functors $F : \mathbf{Prob}^{op} \rightarrow \mathbf{Forest}$ will be a category formed by one single problem.¹ An interesting consequence is that $\mathbf{Forest}^{\bullet op}$ becomes the category \mathbf{Forest} itself. So, in what follows, we will be using the internal logic of the topos of forests to specify (meta)heuristics for one single problem.

Like all slice topoi of topoi of the form \mathbf{Set}^C , the topos \mathbf{AFA} has a natural number object (NNO), which allows for the definition, in the theory of \mathbf{AFA} , of many sets and structures that are found in everyday mathematics. For example, there is an object Q representing the rational numbers; elements of this object will be used below to express discrete probabilities in the definition of stochastic search strategies.

Given a labeled forest F (i.e., an object of \mathbf{AFA}), we will represent the behavior of a stochastic search strategy by a sequence $(S_i)_{i \in \mathbb{N}}$ where each S_i is a set of triples of the form $\langle T, v, q \rangle$, with T a subforest² of F consisting of all nodes visited up to step i , with v the node last visited, and with q a probability. The sequence $(S_i)_{i \in \mathbb{N}}$ is determined by the set S_0 and by a function *step* from the set of all such sets of triples to itself such that $step(S_i) = S_{i+1}$ for all $i > 0$. The sequence $(S_i)_{i \in \mathbb{N}}$ must be such that the set S_0 contains only triples where T is a single-node subforest of F , and where v is that single node. This means that S_0 is actually a collection of possible initial nodes of the search, each node accompanied by a probability. Moreover, for each triple $\langle T', v', q' \rangle$ in S_{i+1} there must exist some triple $\langle T, v, q \rangle$ in S_i such that either $T = T'$, meaning that no new node is visited at step $i + 1$, or T' is an extension of T by exactly the one node v' . Finally, obvious conditions on the values of the probabilities must also hold. All these considerations will be made formal below.

Recall (from the end of Sec. 2) that we intend to describe objects of the slice topos \mathbf{AFA} in the language of $\mathbf{Forest}^{\mathbf{Prob}^{op}}$; so, we must consider pairs of the form $\langle F, \lambda \rangle$ and $\langle T, \theta \rangle$ (corresponding to labeled forests) instead of (unlabeled) forests F and T . Furthermore, given $\langle F, \lambda \rangle$ and $\langle F', \lambda' \rangle$, we have that a morphism $\langle F, \lambda \rangle \xrightarrow{\alpha} \langle F', \lambda' \rangle$ can only be considered if α preserves the labels of F . This is expressed by the predicate³

$$\mathbf{isLabelPreserving}(\alpha, F, \lambda, F', \lambda') \iff \forall x \in F : \lambda(x) = \lambda'(\alpha(x))$$

¹A more thorough treatment, with \mathbf{Prob} an arbitrary thin, skeletal category, is found in [do Amaral 2004].

²Here and in what follows, by a “subforest of F ” we mean a subobject of F , i.e., a forest that can be embedded in the forest F by a forest homomorphism. The roots of the subforest must be at the same level as the roots of the forest.

³The language of LST is typed; here, in order to unclutter the notation, we omit typing information about all terms whose type can be apprehended from the context.

Then, $\langle G, \kappa \rangle$ is a subobject of $\langle F, \lambda \rangle$ iff the following predicate is satisfied (where $\mathbf{isMonic}(\alpha)$ is satisfied iff α is a monomorphism — see [Bell 1988]):

$$\mathbf{isSubObject}(G, \kappa, F, \lambda) \iff \exists \alpha \in F^G : (\mathbf{isMonic}(\alpha) \wedge \mathbf{isLabelPreserving}(\alpha, G, \kappa, F, \lambda))$$

Sometimes, in order to make the formulae more readable, the labeling morphisms (e.g., λ, κ , etc.) and the condition that a morphism α must be label-preserving will be omitted, except where they must be explicitly mentioned. So, labeled forests and subforests will simply be denoted by F, G, T , etc.

The set of all subforests of a given labeled forest F will be denoted PF .

A node G of a labeled forest F is identified with the finite path from the root to the node. In other words, a node G is a nonempty, finite, linear subobject of the forest. Nonemptiness corresponds to G being different from the initial object \emptyset ; finiteness corresponds to the unique morphism 1_G from G to 1 not being epic (where 1 is the terminal object of $\mathbf{Forest}^{\mathbf{Prob}^{op}}$: a linear tree with infinitely many nodes); linearity corresponds to 1_G being monic:

$$\mathbf{isNode}(G, \kappa, F, \lambda) \iff \mathbf{isSubObject}(G, \kappa, F, \lambda) \wedge G \neq \emptyset_F \wedge \mathbf{isMonic}(1_G) \wedge \neg \mathbf{isEpic}(1_G)$$

Recall that L is the codomain of the morphisms responsible for the labeling of nodes (Sec. 2). Then the set of all nodes of a labeled forest $\langle F, \lambda \rangle$ is represented by the following set-term, which we will abbreviate by $nodes(F, \lambda)$:

$$nodes(F, \lambda) = \{ \langle G, \kappa \rangle \in \coprod_{G \in PF} L^G \mid \mathbf{isNode}(G, \kappa, F, \lambda) \}$$

Again, we will omit labeling morphisms and write $nodes(F)$. Clearly, we have that $nodes(F) \subseteq PF$.

A root of a labeled forest F is an element of $nodes(F)$ that is minimal with respect to the partial order “is a subobject of”:

$$\mathbf{isRoot}(G, F) \iff G \in nodes(F) \wedge \neg \exists G' \in nodes(F) : \mathbf{isSubObject}(G', G)$$

The fact that a labeled forest F extends another forest F' by exactly one node is represented by the following predicate:

$$\mathbf{extendsByOne}(F, F') \iff \mathbf{isSubObject}(F', F) \wedge \exists! G \in nodes(F) : G \notin nodes(F')$$

And the following predicate states that the node given by G is the one that was added to F' to yield F :

$$\mathbf{wasAdded}(G, F, F') \iff \mathbf{extendsByOne}(F, F') \wedge G \in nodes(F) \wedge G \notin nodes(F')$$

In our definition of stochastic search strategies, we want to consider only finite, nonempty sets of triples of the form $\langle T, G, q \rangle$ such that the sum of all probabilities q equals 1. To this end, let C be the set

$$C = \left(\coprod_{T \in PF} \text{nodes}(T) \times \{q \in Q \mid 0 < q \leq 1\} \right)$$

where Q is the object of rational numbers. The elements of C are triples of the form $\langle T, G, q \rangle$ with q a probability. Now define S to be the set⁴

$$S = \left\{ X \in PC \mid X \neq \emptyset, X \text{ finite, } \sum_{(x,y,q) \in X} q = 1 \right\}$$

Now we may give a definition of a stochastic search strategy in LST:

Definition 3.1 (Stochastic search strategy). *A stochastic search strategy over a forest F is represented by a term $\langle \text{init}, \text{step} \rangle$ with init a term of type \mathbf{S} , and step a term of type $\mathbf{S}^{\mathbf{S}}$ satisfying the following predicate:*

$$\begin{aligned} & \text{isStochasticSearchStrategy}(\text{init}, \text{step}, F) \iff \\ & \forall \langle T, G, q \rangle \in \text{init} : (T = G \wedge \text{isRoot}(G, F)) \wedge \\ & \forall X, Y \in S : (\text{step}(X) = Y \Rightarrow \\ & \quad \forall \langle T, G, q \rangle \in Y : \exists \langle T', G', q' \rangle \in X : \\ & \quad \quad (T = T' \vee (\text{extendsByOne}(T, T') \wedge \text{wasAdded}(G, T, T')))) \\ &) \end{aligned}$$

The above definition formalizes in LST the description of a stochastic search strategy given at the beginning of this section.

4. A Simple Example: Greedy Search

We illustrate the use of our framework by specifying in LST the rather simple search strategy known as *greedy search*. This is a deterministic search strategy, which means that all triples in C are actually of the form $\langle T, G, 1 \rangle$, representing the fact that, at each iteration, some node G is visited with probability 1. More complex examples involving “truly” stochastic strategies (e.g., simulated annealing, genetic algorithms), which are not included here for lack of space, can be found in [do Amaral 2004].

We need to define two additional useful terms: in many search strategies, the nodes of the search space are evaluated according to some heuristic function. In our language, a heuristic function to evaluate the nodes of a given labeled forest F can be defined as a term h of type $\mathbf{N}^{\text{nodes}(F)}$, where N is the natural number object. In other words, the “grade” a node receives upon evaluation is a natural number. We assume that the lower the grade, the better the evaluation.

For every search strategy $\langle \text{init}, \text{step} \rangle$ we may define by simple recursion (see [Bell 1988]) a term stage of type $\mathbf{S}^{\mathbf{N}}$, where N is the natural number object. The

⁴For space limitations, we do not include the definitions of the “finiteness” predicate or of the summation term. See [do Amaral 2004].

idea is that, given any natural number n , the term $stage(n)$ will be equivalent to $step(step(\dots step(init)\dots))$, with n applications of $step$.

In a simple greedy search strategy, the node visited at each stage after the first is the best child of the node visited in the previous stage, as long as the best child has a better (or equal) evaluation than the node visited in the previous stage. Nothing is assumed about the initial node.

As simple greedy search is deterministic rather than stochastic, for each n the set $stage(n)$ is a singleton. To simplify the formulae, for all n we consider $stage(n)$ to be a pair of the form $\langle T, G \rangle$, with T representing the labeled forest (a tree, actually) formed by the nodes visited so far and G the last node visited.

A search strategy $\langle init, step \rangle$ is considered greedy if it satisfies the predicate in Fig. 1 for some interpretation of h (the heuristic function). There, $children(G)$ is a term denoting the set of children⁵ of node G in forest F . Line 4 specifies that the search terminates (i.e., the current node is revisited indefinitely) when the current node G has no better children. Line 6 says that otherwise the next node to be visited is the best child of the current node.

$$\begin{aligned}
& isGreedy(init, step, F, h) \iff \\
& 1 \quad \mathbf{isStochasticSearchStrategy}(init, step, F) \wedge \\
& 2 \quad \forall T, T', G, G' : (\\
& 3 \quad \quad step(T, G) = (T', G') \iff (\\
& 4 \quad \quad \quad G' = G \wedge \forall K \in children(G, F) : h(K) > h(G) \\
& 5 \quad \quad \quad \vee \\
& 6 \quad \quad \quad h(G') \leq h(G) \wedge \forall G'' \in children(G, F) : h(G') \leq h(G'') \\
& 7 \quad \quad) \\
& 8 \quad)
\end{aligned}$$

Figure 1. Specification of greedy search in LST

By using the sound and complete sequent calculus for Local Set Theory defined in [Bell 1988], one can prove properties of the search strategies specified in our framework. We offer below some brief comments on examples of provable formulae involving greedy search. The properties are quite simple and intuitive, and are included here only for illustrative purposes. The proofs themselves are not presented, and although a bit lengthy, can be easily constructed.

The provable formula that says that greedy search is deterministic (i.e., not stochastic) asserts that each set $stage(n)$ is a singleton:

$$\begin{aligned}
& \forall init, step, F, h : \mathbf{isGreedy}(init, step, F, h) \implies \\
& \quad \forall n : \exists! \langle T, G \rangle : \langle T, G, 1 \rangle \in stage(n)
\end{aligned}$$

Furthermore, greedy search never backtracks. Equivalently, at each iteration, the subforest consisting of all nodes visited by the search is actually a linear tree. This is

⁵A node G' is a child of a node G iff $\mathbf{extendsByOne}(G', G)$ is satisfied.

expressed by the following provable formula (recall that a node is identified with a path from the root to the node; i.e., a finite linear tree):

$$\forall \text{init}, \text{step}, F, h : \mathbf{isGreedy}(\text{init}, \text{step}, F, h) \Rightarrow \\ \forall n : \forall \langle T, G, q \rangle \in \text{stage}(n) : \mathbf{isNode}(T)$$

The implementation of theorem-proving and model-checking techniques for LST is work in progress. As soon as this is completed, we will be able to present examples of the verification of more complex, real-life properties of search strategies.

5. Conclusion

Related work. Our use of category- and topos-theoretical tools and techniques to build frameworks for problems and (meta)heuristics is, to the best of our knowledge, unprecedented. In [Blass 1995], different categories of problems and reductions (“questions and answers”) are examined, and — somewhat surprisingly — found to be present in subjects as diverse as linear logic and cardinal characteristics of the continuum. Our category **Prob** of problems is a minor variation of one of these categories of questions and answers. A deeper study of **Prob** and related categories is beyond the scope of this article, but can be found in [do Amaral 2004]. In [Leal et al. 2001], categories of optimization problems and reductions are defined, and approximation algorithms are examined from a categorical perspective, but no topos-theoretical considerations are made, and the approach is not logic-based.

Further research. Deeper considerations about our framework can be found in [do Amaral 2004]. Future work includes the development of theorem-proving and model-checking capabilities for our framework, as well as the application of these capabilities to the specification and verification of (meta)heuristic search algorithms found in practice.

References

- Bell, J. L. (1988). *Toposes and Local Set Theories, an Introduction*. Oxford U. Press.
- Blass, A. (1995). Questions and answers: a category arising in linear logic, complexity theory, and set theory. In *Proceedings of the Workshop on Advances in Linear Logic*, pages 61–81. Cambridge University Press.
- do Amaral, F. N. (2004). *Teoria de Modelos para Heurísticas Baseada em Topoi (in Portuguese)*. PhD thesis, PUC-Rio, Brazil. Available at http://www2.dbd.puc-rio.br/arquivos/145000/147500/10_147591.htm.
- Goldblatt, R. (1979). *Topoi – the Categorical Analysis of Logic*. North Holland.
- Leal, L. A. S., Menezes, P. B., Claudio, D. M., and Toscani, L. V. (2001). Optimization problems categories. In Moreno-Diaz, R. and Quesada-Arencibia, A., editors, *EUROCAST 2001 – Extended Abstracts*, pages 93–96.
- Pearl, J. (1985). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. The Addison-Wesley Series in Artificial Intelligence. Addison-Wesley.
- Voss, S. (2001). Meta-heuristics: The state of the art. In *Proceedings of the Workshop on Local Search for Planning and Scheduling*, pages 1–23. Springer-Verlag.