Up-to-Isomorphism Enumeration of Finite Models - The Monadic Case

Thierry Boy de la Tour*
LEIBNIZ - IMAG

1 Introduction

Few systems for finite model enumeration use pruning techniques based on the notion of isomorphism between finite interpretations (see [2],[1]), known to correspond to elementary equivalence of interpretations. These pruning techniques have two important restrictions: only special isomorphisms are considered, and only the last interpretation which has not been eliminated this way is tested for isomorphism with new candidate interpretations, which may be isomorphic with previous interpretations. The first point is due to the fact that there are many isomorphisms ($n!$ for an interpretation of size n), isomorphism testing is time consuming, and it would be pointless to spend more time on such tests than would require the evaluation of the formula on the candidate interpretation. This is even more likely to happen were all the previous interpretations to be tested for isomorphism. Hence only a very limited amount of isomorphism testing may be allowed, and it may appear hopeless to save time by avoiding double work in this context.

On the other hand, we should consider the fact that the amount of double work may be huge, since every interpretation can have up to $n!$ isomorphic interpretations. Which is to say that the number of isomorphism classes (IC) of interpretations may be far less than the number of interpretations. This is ground for attempting a direct enumeration of these IC, which would therefore avoid (as much as possible) the actual *testing* for isomorphism.

More precisely, our aim is not to build explicitely the IC of interpretations one after another, but to enumerate objects representative of each IC, and such that at least one element of the IC can be efficiently computed from this object. This is performed by an analysis of the structure of the IC of first-order objects (functions and relations, see section 2 for the analysis of monadic objects) and a group-theoretic study of how the IC of interpretations relate with the IC of objects of which they are composed (section 4).

The up-to-isomorphism enumeration appear to be quite complex, though some similarities occur at different levels of the enumeration, which makes it very convenient to design the procedure as a so-called "Successor Machine" (see section 3), constructed by means of elementary general-purpose successor functions. An important requirement of the enumeration is the possibility to switch to sub-enumeration ensuring minimal changes with the next interpretation (this is the notion of *refutation in an interpretation* from [1]), and we give some promising hints about the application of this idea to our framework in section 5. Since the structure of IC of functions are very different whether image and range sets are equal or not, we adopt a many-sorted account of finite semantics of first-order logic.

*46, avenue Félix Viallet, 38031 Grenoble Cedex, FRANCE, Thierry.Boy-de-la-Tour@imag.fr

We are given a finite set $\mathfrak{T}$ of basic types, or *sorts*, from which we build the set of *first-order types* $\mathfrak{T}_1 = \bigcup_{k\in\mathbb{N}} \mathfrak{T}^k \times (\mathfrak{T} \uplus \{\mathbf{o}\})$. They are noted $t_1 \times \ldots \times t_k \to t$, and said to be *homogenous* if $t_1 = \ldots = t_k = t$. A $\mathfrak{T}$-*signature* is a finite sequence $\Sigma = (\tau_1, \ldots, \tau_n)$ of elements of $\mathfrak{T}_1$ (intended as the types of the symbols $s_1 \ldots s_n$ with which the formulas are written). We are also given a function $\mathfrak{n} : \mathfrak{T} \to \mathbb{N}^*$, in order to interpret sorts $t \in \mathfrak{T}$ as $t^{\mathfrak{n}} = \{1_t, \ldots, \mathfrak{n}(t)_t\}$ (integers indexed by t so that $t \neq t' \Rightarrow t^{\mathfrak{n}} \cap t'^{\mathfrak{n}} = \emptyset$.) Then $\mathbf{o}^{\mathfrak{n}} = \{\top, \bot\}$ are the truth values and $(t_1 \times \ldots \times t_k \to t)^{\mathfrak{n}}$ is the set of functions from $t_1^{\mathfrak{n}} \times \ldots \times t_k^{\mathfrak{n}}$ to $t^{\mathfrak{n}}$. The set of $\mathfrak{n}$-Σ-*interpretations* is $\Sigma^{\mathfrak{n}} = \prod_{i=1}^{n} \tau_i^{\mathfrak{n}}$.

Let $\mathfrak{G} = \prod_{t\in\mathfrak{T}} \mathrm{Sym}\ t^{\mathfrak{n}}$ be the set of $\mathfrak{n}$-$\mathfrak{T}$-isomorphisms ($\mathfrak{G}$ is the biggest permutation group on $\biguplus_{t\in\mathfrak{T}} t^{\mathfrak{n}}$ which stabilizes $t^{\mathfrak{n}}$ for all $t \in \mathfrak{T}$). $\forall \tau \in \mathfrak{T}_1$, we define the operation of $\mathfrak{G}$ on $\tau^{\mathfrak{n}}$ as follows: $\forall f \in (t_1 \times \ldots \times t_k \to t)^{\mathfrak{n}}, \forall \sigma \in \mathfrak{G}, \forall \langle i_1, \ldots, i_k\rangle \in t_1^{\mathfrak{n}} \times \ldots \times t_k^{\mathfrak{n}}, f^\sigma(i_1\sigma, \ldots, i_k\sigma) = f(i_1, \ldots, i_k)\sigma$ (i.e. $f^\sigma \in \tau^{\mathfrak{n}}$ is obtained by applying σ to the graph of $f \in \tau^{\mathfrak{n}}$.) Remember that $\mathbf{o}$ is not a sort, hence $\top\sigma = \top$ and $\bot\sigma = \bot$. The operation of $\mathfrak{G}$ on $\mathfrak{n}$-Σ-interpretations is a standard extension of the previous one: $\forall \langle f_1, \ldots, f_n\rangle \in \Sigma^{\mathfrak{n}}, \langle f_1, \ldots, f_n\rangle^\sigma = \langle f_1^\sigma, \ldots, f_n^\sigma\rangle$. Hence $\forall \Im, \Im' \in \Sigma^{\mathfrak{n}}$, $\Im$ and $\Im'$ are isomorphic iff $\Im^{\mathfrak{G}} = \Im'^{\mathfrak{G}}$, where $\Im^{\mathfrak{G}} = \{\Im^\sigma / \sigma \in \mathfrak{G}\}$ is the $\mathfrak{G}$-*orbit* of $\Im$, hence IC may be considered as orbits, and our aim is to enumerate the orbit partition $\mathrm{OPart}(\Sigma^{\mathfrak{n}}, \mathfrak{G}) = \{\Im^{\mathfrak{G}} / \Im \in \Sigma^{\mathfrak{n}}\}$.

2 $\mathfrak{G}$-orbits of monadic functions

2.1 monadic relations

Let $\tau = t \to \mathbf{o}$ with $t \in \mathfrak{T}$. $\forall R \in \tau^{\mathfrak{n}}$, let $c(R) = \mathrm{card}(R^{-1}(\top))$. We have $0 \leq c(R) \leq \mathfrak{n}(t)$, and from all $k \in \{0, \ldots, \mathfrak{n}(t)\}$ we may build a relation $R_k \in \tau^{\mathfrak{n}}$ such that $c(R_k) = k$ (for instance $R_k(i_t) = \top$ iff $i \leq k$.) It is very easy to prove that $\forall \sigma \in \mathfrak{G}, c(R^\sigma) = c(R)$, as well as the following converse: $\forall R, R' \in \tau^{\mathfrak{n}}$, if $c(R) = c(R')$ then we may build a $\sigma \in \mathfrak{G}$ such that $R^\sigma = R'$. This means that $R^{\mathfrak{G}} = R'^{\mathfrak{G}}$ (R and R' are isomorphic) iff $c(R) = c(R')$. Hence enumerating $\mathrm{OPart}(\tau^{\mathfrak{n}}, \mathfrak{G})$ only requires the enumeration of the integers $k \in \{0, \ldots, \mathfrak{n}(t)\}$ and the computation of the relations R_k. This enumeration is exact since no orbit is visited twice, i.e. $\mathrm{OPart}(\tau^{\mathfrak{n}}, \mathfrak{G}) = \biguplus_{k=0}^{\mathfrak{n}(t)} \{R_k^{\mathfrak{G}}\}$.

2.2 heterogenous functions

Let $\tau = t \to t'$ with $t, t' \in \mathfrak{T}, t \neq t'$. $\forall f \in \tau^{\mathfrak{n}}$, let $p(f) = [\mathrm{card}(f^{-1}(i))/i \in t'^{\mathfrak{n}} \wedge f^{-1}(i) \neq \emptyset]$ (square brackets are for multi-sets.) We clearly have $1 \leq \mathrm{card}(p(f)) \leq \mathfrak{n}(t')$ and $\sum p(f) = \mathfrak{n}(t)$, hence $p(f)$ is a partition of $\mathfrak{n}(t)$ with at most $\mathfrak{n}(t')$ summands (a $\mathfrak{n}(t')^-$-partition.) Given such a partition P, it is easy to build a function $f_P \in \tau^{\mathfrak{n}}$ such that $p(f_P) = P$. As above, we have $\forall f, f' \in \tau^{\mathfrak{n}}, f^{\mathfrak{G}} = f'^{\mathfrak{G}}$ iff $p(f) = p(f')$. Hence we obtain an exact enumeration of $\mathrm{OPart}(\tau^{\mathfrak{n}}, \mathfrak{G})$ from any exact enumeration of the set of $\mathfrak{n}(t')^-$-partitions of $\mathfrak{n}(t)$. We may begin with the 1-partition $[\mathfrak{n}(t)]$, then the 2-partitions starting from $[\mathfrak{n}(t) - 1, 1]$ and ending with $[E(\frac{\mathfrak{n}(t)+1}{2}), E(\frac{\mathfrak{n}(t)}{2})]$, etc.

2.3 homogenous functions

Let $\tau = t \to t$ with $t \in \mathfrak{T}$. It is standard in the context of finite interpretations to represent functions $f \in \tau^{\mathfrak{n}}$ as sequences $\langle f(1_t), \ldots, f(\mathfrak{n}(t)_t)\rangle$, but we may as well represent them as directed graphs $G(f)$ with set of vertices $t^{\mathfrak{n}}$ and set of edges $\{\langle i, f(i)\rangle / i \in t^n\}$. We then interpret isomorphisms between functions as graph isomorphisms ($f^{\mathfrak{G}} = f'^{\mathfrak{G}}$ iff $G(f)$ and

$G(f)$ are isomorphic.) Clearly, the graphs obtained this way have output degree one, which means that every connex component contains a cycle (paths have no ends, but the number of vertices is finite), and that the graphs connected to this cycle can only be trees. It is easy to see that the graph isomorphism problem on these monadic graphs is polynomial. Providing an efficient non-redundant (up to isomorphism) enumeration of monadic graphs of a given size is however not a trivial task. Next section, this is profitably decomposed into the problems of enumerating connex components, cycles, "cyclic-graftings" and trees.

3 The Successor Machine

The principle of the successor machine stems from additioner circuits. It can be seen as a machine having (at least) an input signal `succ` and an output signal `carry`. It also has an internal state, coding an element in a finite linear order. Each time the `succ` signal is activated, the machine changes its internal state to the code of the successor (in the linear order) of the previously encoded element, unless it was the maximal element, in which case the machine activates its output `carry` signal instead. For instance, a bit is a successor machine with two possible states 0 and 1. In order to compute successors of n-bits integers, we may build a machine CP with (apart from `succ` and `carry`) outputs $\texttt{succ}_1 \dots \texttt{succ}_n$ plugged in the `succ` inputs of the bits $m_1 \dots m_n$, and inputs $\texttt{carry}_1 \dots \texttt{carry}_n$ coming from $m_1 \dots m_n$. The internal structure of CP is very simple: `succ` is connected to $\texttt{succ}_1$, `carry` to $\texttt{carry}_n$ and $\forall i \in \{1, \dots, n-1\}$, $\texttt{carry}_i$ is connected to $\texttt{succ}_{i+1}$. Provided each bit comes back to its 0 state when it sends a `carry` signal, the machine $CP(m_1, \dots, m_n)$ enumerates the n-bits integers in the usual order. The machine CP actually enumerates the Cartesian product of the sets enumerated by its sub-machines, in the lexicographic order. The real machine is a bit more complex, it sends and receives some `zero` signals, etc. and is programmed in object-oriented style.

But we need more complex products in order to perform an up-to-isomorphism enumeration of monadic graphs. For instance, if $C_2 = \{a, b\}$ is a set of representants of the orbits of connex monadic graphs of size 2, then an enumeration of the monadic graphs of size 4 may contain $\langle a, a\rangle, \langle a, b\rangle, \langle b, b\rangle$, skipping $\langle b, a\rangle$ from the Cartesian product $C_2 \times C_2$ since it is isomorphic to $\langle a, b\rangle$. More generally, considering k sets $S_1, \dots, S_k$ either equal or disjoint two by two, and a subgroup $G <$ Sym k such that $\forall \sigma \in G, \forall i \in \{1 \dots k\}, S_{i\sigma} = S_i$, then a *$G$-product* of $S_1, \dots, S_k$ is any minimal $P \subset S_1 \times \dots \times S_k$ verifying: $\forall \langle x_1, \dots, x_k\rangle \in S_1 \times \dots \times S_k, \exists \sigma \in G, \langle x_{1\sigma}, \dots, x_{k\sigma}\rangle \in P$. We are actually interested in the *symmetric product* $\prod^{\mathrm{S}}$, which is the G-product with the biggest possible $G <$ Sym k (according to the relations among the S_i's), and the *circular product* $\prod^{\mathrm{C}}$, obtained with the biggest possible $G < < (1, 2, \dots, k) >$ (cyclic group of order k). Hence the set enumerated above is the symmetric product $C_2 \times^{\mathrm{S}} C_2$. Trees can also be enumerated this way: if $\mathcal{T}_n$ is a set of representants of IC of trees of size n, we have if $n > 1$

$$\mathcal{T}_n = \biguplus_{1 \le k \le n-1} \ \biguplus_{p \in \mathrm{Part}_k(n-1)} \ \prod_{1 \le i \le k}^{\mathrm{S}} \mathcal{T}_{p_i}$$

where k is the (input) degree of the root, $n-1$ the total size of the k subtrees, $\mathrm{Part}_k(n-1)$ is the set of k-partitions of $n-1$ (to be enumerated by a successor machine), and p_i the i^{th} element of p considered as a list. Similarly, representants of IC of monadic graphs of size

n can be enumerated by the successor machine corresponding to the following expression:

$$\biguplus_{1\le k\le n}\ \biguplus_{p\in \mathrm{Part}_k(n)}\ \prod^{\mathrm{s}}_{1\le i\le k}\ \biguplus_{1\le c\le p_i}\ \biguplus_{q\in \mathrm{Part}_c(p_i)}\ \biguplus_{q'\in \mathrm{CL}(q)}\ \prod^{\mathrm{c}}_{1\le j\le c} \mathcal{T}_{q'_j}$$

where k is the number of connex components, the i^{th} having size p_i and containing a cycle of size c, on which are grafted c trees of sizes $q_1,\ldots,q_c$ in the order given by the lists $q'\in \mathrm{CL}(q)$, enumerating all possible permutations of q up to cyclic permutations.

One feature necessary for the symmetric and circular products is the possibility to perform assignment on coordinates (the internal state of one machine is set to the internal state of another). It is however remarkable that no equality testing is necessary, and that none of these elementary successor machines needs to keep track of passed work, hence performing a space-less search. A prototype implementation of the machine for monadic graphs has been programmed in OCAML. Here is a summary of the results (the last column yields the average number of automorphisms of the corresponding graphs.)

n	nb of functions ($=n^n$)	$m=$ nb of orbits	$\frac{mn!}{n^n}$
5	$3,125$	47	1.805
10	$10,000,000,000$	$7,318$	2.655
15	$437,893,890,380,859,375$	$1,328,993$	3.9687
20	$1,048,576*10^{20}$	$258,604,642$	6.0001

4 $\mathfrak{G}$-orbits of $\mathfrak{n}$-Σ-interpretations

For example, consider the orbit of constant functions $\{\bar{i}/i\in t^{\mathfrak{n}}\}=\bar{1_t}^{\mathfrak{G}}$, and the single orbit of a t-constant $1_t^{\mathfrak{G}}=t^{\mathfrak{n}}$. In order to enumerate $\mathrm{OPart}(\Sigma^{\mathfrak{n}},\mathfrak{G})$ for $\Sigma=\langle t\to t,t\rangle$, we have to enumerate the product $\bar{1_t}^{\mathfrak{G}}\times 1_t^{\mathfrak{G}}$, and it is clear that $\langle\bar{1_t},1_t\rangle^{\mathfrak{G}}$ is only a subset of it; we also need $\langle\bar{1_t},2_t\rangle^{\mathfrak{G}}$ (or $\langle\bar{2_t},1_t\rangle^{\mathfrak{G}}$). Hence $\mathrm{OPart}(\Sigma^{\mathfrak{n}},\mathfrak{G})$ contains much more elements than the product of the $\mathrm{OPart}(\tau_i,\mathfrak{G})$. We can prove that $\mathrm{OPart}(\Sigma^{\mathfrak{n}},\mathfrak{G})$ is equal to:

$$\biguplus_{o_1\in \mathrm{OPart}(\tau_1^{\mathfrak{n}},\mathfrak{G}^{(1)})}\ \biguplus_{o_2\in \mathrm{OPart}(\tau_2^{\mathfrak{n}},\mathfrak{G}^{(2)})}\ \cdots\ \biguplus_{o_n\in \mathrm{OPart}(\tau_n^{\mathfrak{n}},\mathfrak{G}^{(n)})}\ \{\langle\overline{o_1},\overline{o_2},\ldots,\overline{o_n}\rangle^{\mathfrak{G}}\}$$

where $\overline{o_i}$ is any element of o_i, $\mathfrak{G}^{(1)}=\mathfrak{G}$ and $\mathfrak{G}^{(i+1)}=\mathfrak{G}^{(i)}_{\overline{o_i}}=\{\sigma\in\mathfrak{G}^{(i)}/\overline{o_i}^{\sigma}=\overline{o_i}\}$ is the *stabilizer* of $\overline{o_i}$ in $\mathfrak{G}^{(i)}$. The fact that these unions are disjoint means that we can provide an exact enumeration of $\mathrm{OPart}(\Sigma^{\mathfrak{n}},\mathfrak{G})$ whenever we are able to provide exact enumerations of $\mathrm{OPart}(\tau_i^{\mathfrak{n}},\mathfrak{G}^{(i)})$, and also that the number of representants computed this way does not depend on the order in which the τ_i's are ordered in Σ, though obviously the $\mathfrak{G}^{(i)}$'s do.

Hence the enumeration of $\mathrm{OPart}(\tau^{\mathfrak{n}},\mathfrak{G})$ sketched above for monadic types does not seem to be general enough. However, a direct enumeration of (and the design of a successor machine for) $\mathrm{OPart}(\tau^{\mathfrak{n}},\mathcal{G})$ for any given $\mathcal{G}<\mathfrak{G}$ seems hopelessly tricky, and we rather adopt a general solution stemming from the fact that $\mathrm{OPart}(\tau^{\mathfrak{n}},\mathcal{G})$ is a refinement of $\mathrm{OPart}(\tau^{\mathfrak{n}},\mathfrak{G})$. Hence any $\mathcal{G}$-orbit can be obtained as $(\overline{o}^{\sigma})^{\mathcal{G}}$ where $o\in\mathrm{OPart}(\tau^{\mathfrak{n}},\mathfrak{G})$ and $\sigma\in\mathfrak{G}$. Since $o=\bigcup_{\sigma\in\mathfrak{G}}(\overline{o}^{\sigma})^{\mathcal{G}}$, the problem is to find a suitable subset $\{\sigma_1,\ldots,\sigma_m\}\subset\mathfrak{G}$ in order to make this union a disjoint one, hence obtaining an exact enumeration of $\mathrm{OPart}(\tau^{\mathfrak{n}},\mathcal{G})$. It is not difficult to prove that $\{\mathfrak{G}_{\overline{o}}\sigma_i\mathcal{G}/1\le i\le m\}$ should be a partition of $\mathfrak{G}$, hence computing the σ_i's is a double coset enumeration problem, well studied in computational group theory, which provides efficient algorithms for computing such double

coset representatives. This is also true for the computation of stabilizers, and we are therefore able to perform an exact enumeration of $\mathrm{OPart}(\Sigma^{\mathfrak{n}}, \mathfrak{G})$ from exact enumerations of $\mathrm{OPart}(\tau^{\mathfrak{n}}, \mathfrak{G})$.

5 Successors with minimal changes

Not all values set by an interpretation $\Im$ may be necessary to evaluate the formula to $\bot$. Hence the evaluation process can easily yield a partial interpretation $\Im'$ such that all interpretations having $\Im'$ as sub-interpretation are counter-models of the formula (see [1] for details). Hence from $\Im$ the enumeration may skip these interpretations, and reach directly the smallest interpretation bigger than $\Im$ which changes at least one value set by $\Im'$. In our context, we may skip them if at least one member in their IC contains $\Im'$, hence this sub-interpretation relation should be interpreted up-to-isomorphism.

The case of monadic heterogenous functions is particularly interesting. Since from any $f \in (t \to t')^{\mathfrak{n}}$ can be computed $p(f)$, a sub-interpretation of f can be interpreted as a sub-partition P of $p(f)$ as well, i.e. such that $\mathrm{card}(P) \leq \mathrm{card}(p(f))$ and $\exists \pi$ a 1-1 function (on multi-sets!) from P to $p(f)$ such that $\forall i \in P, i \leq \pi(i)$. An efficient program has been written in OCAML which, when given a partition $p(f)$ and a sub-partition P, yields the smallest partition (in the order sketched above) of which P is not a sub-partition, or sends `carry` if there is none. An interesting application is the pigeon hole formula, which states that a monadic heterogenous function is 1-1. When a finite model is searched for with $\mathfrak{n}(t) = n + 1$ (pigeons) and $\mathfrak{n}(t') = n$ (holes), the evaluation on the first interpretation f yields $\bot$ and as sub-interpretation two pigeons $i, j \in t^{\mathfrak{n}}$ such that $f(i) = f(j)$. Hence $P = [2]$, and the next partition must not contain any $a \geq 2$. Since there is no n^--partition of $n + 1$ containing only ones, our program sends the `carry` signal, and the search is over.

This is more difficult with monadic homogenous functions, since sub-interpretations are subgraphs, and the problem of testing for inclusion of a forest in a tree is known to be NP-Complete. We may however find some partial solutions, for instance if a connex component receives a `succ` while only, but not all, values of its cycle have been used, then it can send a `carry`, since in our enumeration the length of cycles increases.

6 Conclusion

The successor machine clearly helps the design of enumerating rather complex structures such as IC of monadic first-order interpretations. This structure is however simple enough that isomorphism testing is still polynomial. This is no longer the case of non-monadic signatures (ruling out the equality), for which it is clearly isomorphism complete. It is doubtful whether a polynomial successor function can then exist, and it may be necessary for efficiency requirements to allow for some redundancy in the search.

References

[1] Nicolas Peltier. A new method for automated finite model building exploiting failures and symmetries. *Journal of Logic and Computation*, To appear.

[2] Jian Zhang and Hantao Zhang. Sem: a system for enumerating models. In C. S. Mellish, editor, *Proceedings of IJCAI'95*, volume 1, pages 298–303, Montréal, august 20-25 1995. Morgan Kaufmann.