Phylogenetic Trees in ACL2

Warren A. Hunt Jr.
Department of Computer Sciences
The University of Texas at Austin
hunt@cs.utexas.edu

Serita M. Nelesen
Department of Computer Sciences
The University of Texas at Austin
serita@cs.utexas.edu

ABSTRACT

Biologists studying the evolutionary relationships between organisms use software packages to solve the computational problems they encounter. Several of these problems involve the production and analysis of phylogenetic trees. We present our system for phylogenetic tree manipulation, TASPI, which was designed to allow the specification and verification of various tree operations, while giving good execution performance. The first aspect of TASPI is a new format for storing and manipulating phylogenetic trees that significantly reduces storage requirements while continuing to allow the trees to be used as input to post-tree analysis. We also prove the correspondence of this format to another tree format. In addition, we give a consensus algorithm with verified guards that is faster by an order of magnitude than standard phylogenetic analysis tools.

Categories and Subject Descriptors

E.2 [**Data Storage Representations**]: Hash-table representations; F.4.1 [**Mathematical Logic**]: Mechanical theorem proving; D.2.4 [**Software/Program Verification**]: Correctness proofs

General Terms

Verification, Performance

Keywords

ACL2, Phylogenetics

1. INTRODUCTION

Phylogenetic trees represent the evolutionary relationship between species. Biologists attempt to find the correct phylogenetic tree to answer scientific questions ranging from "Why does Columbia have more types of birds than any other country?" to "Do wings, flippers and arms all evolve from the same structure in some ancient organism?" Phylogenetic trees are also used to answer more practical questions. For example, phylogenetics have been used to determine the sources of infection from HIV [11], and to predict gene function, which is used for drug discovery [13].

ACL2 '06 Seattle, Washington USA

Given the importance of the results of these studies, we would like to know that the software producing the trees in accurate. Unfortunately, this is not currently the case. For example, we found several cases where PAUP [14], one of the most widely used packages for phylogenetic inference, produced output that was not even well-formed (the author was notified of the bug).

This paper introduces TASPI, an experimental system written from scratch in the ACL2 [12] formal logic, that performs a few of the operations used in phylogenetics. In particular, we are addressing those operations that use trees as input. Our goal is to build a system that is clearly specified, and about which various properties can be proved. In addition, we would like this system to be competitive in terms of execution speed.

Towards these goals, we have developed methods for storing and retrieving phylogenetic tree data, and using these methods we have implemented a consensus algorithm. Our approach permits very large data sets to be compactly stored and retrieved without any loss of precision. Also, our implementation of our consensus algorithm provides greatly increased performance when performing strict and majority consensus computations as compared to currently available software, such as PAUP [14] and TNT [9].

2. BACKGROUND

Biologists interested in the phylogenetic relationship between organisms attempt to produce a tree that best represents evolutionary history. Given that the true history can never be known for certain, they instead produce a tree that optimizes some criterion (with information about the species involved at the leaves). Two of the most common criteria for deciding which tree is the correct tree are Maximum Parsimony (MP) and Maximum Likelihood (ML).

Both MP and ML are computationally intensive (they have been proven to be NP-hard [5] [4]). Since the number of possible trees grows extremely fast as the number of taxa (species) increases, heuristics are used to attempt to find the optimal tree without doing an exhaustive search. This can result in large numbers of trees with equally (or nearly equal) optimal scores.

When a biologist is presented with multiple trees that are equally good as far as their optimization criteria is concerned, they often perform some form of consensus analysis to create a consensus tree. Consensus trees are defined by Felsenstein as "trees that summarize, as nearly as possible, the information contained in a set of trees whose tips are all the same species" [8]. Consensus methods return a single tree, or an indication that no tree meeting that method's requirements exists.

There are many types of consensus trees, but two of the most common are strict and majority. Both of these decide which branches in the input trees to keep, and then build a tree from the resulting branches. Strict consensus requires that any branch in the consen-

sus tree be a branch in every input tree, while a majority tree only requires that any branch in the consensus tree be a branch in at least some majority of the input trees. Strict and majority consensus algorithms always return a tree, and have optimal O(kn) algorithms as described by Day [6] and Amenta et al. [1] (where k is the number of trees and n is the number of taxa).

3. REPRESENTATIONS

3.1 TASPI's tree format

Newick format [7] is the standard way of storing a collection of phylogenetic trees. Adopted in 1986, Newick is a parenthetical notation that uses commas to separate sibling subtrees, parentheses to indicate children, and a semicolon to conclude a tree. Newick outlines each tree in its entirety whether storing one tree, or a collection of trees.

On the other hand, our system, TASPI, capitalizes on common structure within a collection of trees. TASPI stores a common subtree once, and then each further time the common subtree is mentioned, TASPI references the first occurrence. This saves considerable space since potentially large common subtrees are only stored once, and the references are much smaller (for empirical results see Section 5).

There are two layers to the TASPI representation of trees. At a high-level, trees are represented as Lisp lists, similar in appearance to Newick, but without commas and semicolons. This is the format presented to the user of TASPI and on which user functions operate. At a low-level, the data are instead represented in a form that uses hash-consing [10] to achieve decreased storage requirements and improved accessing speeds.

We have a couple ACL2 predicates that together recognize well-formed trees. First, **taspip** checks that a tree is made up of symbols and integers for leaves (though `nil` is not a valid taxa name), and that any time there are sibling subtrees, they are part of a **true-listp**.

```
(defun taspip (flg tree)
  (if flg
      (if (atom tree)
          (or (and (symbolp tree)
                   (not (equal tree nil)))
              (integerp tree))
        (taspip nil tree))
    (if (atom tree)
        (null tree)
      (and (taspip t (car tree))
           (taspip nil (cdr tree))))))
```

Second, **ordered-taspi** checks that a tree follows a specified ordering. Any sibling taxa are to be in the order given, and sibling subtrees must have the first taxa in each of their representations in the correct order.

```
(defun first-taxon (term)
  (if (atom term)
      term
    (first-taxon (car term))))

(defun ordered-taspi (tree order)
  (if (consp tree)
      (if (consp (cdr tree))
          (and (< (cdr
                    (assoc-equal
                      (first-taxon tree)
                      order))
                  (cdr
                     (assoc-equal
                       (first-taxon
                          (cdr tree))
                       order)))
               (ordered-taspi (car tree)
                              order)
               (ordered-taspi (cdr tree)
                              order))
        (ordered-taspi (car tree) order))
    t))
```

3.2 Trees from bipartitions

Lisp programmers can easily see how the parenthetical notation given above is intuitive for representing trees. However, for several algorithms using trees as input, having a representation more directly related to the branches of the tree is useful.

A branch in a tree separates the taxa into two sets: the taxa reachable from one side of the branch versus the taxa reachable from the other. These two sets together make up a bipartition. A tree can then be represented by listing the set of bipartitions in the tree.

We represent bipartitions in TASPI by listing the taxa on one side of a branch, and then using a list of all the taxa in the tree to infer the other side. A valid bipartition is recognized by `good-partp`, which checks that there are no duplicate taxa names, that each taxa name is either a symbol or integer, and that there are at least 2 taxa names in the bipartition.

```
(defun good-partp (x)
  (and (int-symlist x)
       (no-duplicatesp-equal x)
       (< 1 (len x))))
```

In order to move between a bipartition representation of trees and the parenthetical notation, we have functions that transform one representation into the other:

```
(defun fringes (flg tree order)
  (if flg
      (if (consp tree)
          (cons
             (ofringe t tree order)
             (append
                 (fringes
                     t (car tree) order)
                 (fringes
                     nil (cdr tree) order)))
        nil)
    (if (consp tree)
        (append (fringes
                    t (car tree) order)
                (fringes
                    nil (cdr tree) order))
      nil)))


(defun partstotaspi (top under order)
  (if (consp under)
      (orderly-cons
         (partstotaspi
             (car under)
             (get-subsets (car under)
                          (cdr under))
             order)
         (partstotaspi
```

```
            (difference top (car under))
            (get-non-subsets (car under)
                             (cdr under))
           order)
       order)
  top))
```

fringes takes a tree in parenthetical notation, and produces a list of bipartitions where each bipartition has been ordered according to the given order. If given a non-nil flag, the first element of the list is not a bipartition, but instead an ordered list of the taxa in the tree. **partstotaspi** takes a list giving all taxa in the tree (top), the list of bipartitions (under) and an ordering in which to create the parenthetical notation.

These functions are not tail-recursive, and as such are not the most efficient implementation. However, these simplified definitions allow us to prove the following theorem:

```
(defthm fringes-partstotaspi-inverse
  (implies
     (and (ordered-taspi x order)
          (good-order-list order)
          (not (and flg
                    (not (consp x))))
          (no-duplicatesp-equal
            (strip-cdrs order))
          (no-duplicatesp-equal (mytips x))
          (subset (mytips x)
                  (get-taxa-from-order order))
          (taspip flg x))
     (if flg
         (equal (partstotaspi
                  (car (fringes flg x order))
                  (cdr (fringes flg x order))
                  order)
                x)
       (equal (partstotaspi
                  (ofringe flg x order)
                  (fringes flg x order)
                  order)
              x)))
  :rule-classes :nil)
```

This theorem says that if given a good ordering of taxa in the tree, and a tree that is in a good form and appropriately ordered, then applying **partstotaspi** to the result of **fringes** gives back the original tree.

4. CONSENSUS ANALYSIS

Given a collection of phylogenetic trees, there are several operations that a biologist may which to perform. The first that we have implemented are majority and strict consensus. We compute a consensus through a sequence of steps. We first read the source file containing the trees for which a consensus is to be computed. During the read process, we identify every subtree for which we have already read an identical subtree. We next create a mapping from all subtrees to their parents. Using this information, we compute the occurrence frequency of every branch. Finally, after we have selected the branches that match our selection criteria, we construct the consensus answer. For a more thorough explanation, see our previous paper [3].

The functions implementing our consensus algorithm have all of their guards verified, which gives us a formal specification for the algorithm. Also, though we have used somewhat simplified versions of the functions for the theorem in Section 3.2, the efficient implementation makes use of BDDs for improved set computations. Further, we have built this system in a version of ACL2 with hash-consing and memoization [2] which we use to avoid recomputation whenever possible.

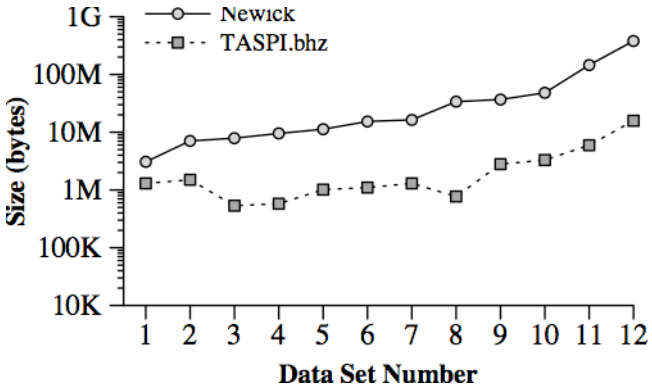

Figure 1: Storage requirements

5. EXPERIMENTS

One of our goals with TASPI is to create a system for phylogenetic trees that is both clearly specified, and that performs well. Having specified our system, we compared its performance to currently available software. We obtained collections of phylogenetic trees generated by researchers, and also generated two large collections ourselves.

Using PAUP [14], TNT [9] and TASPI, we measured the time it took the software to read each collection, and the time needed to compute both a strict and majority consensus tree. We also compared the storage space requirements for each collection in several different formats.

Figure 1 shows two sizes for each of our benchmark data sets. The Newick line represents the size of the file storing trees in Newick format. TASPI.bhz displays the size of the file after compression using the Boyer-Hunt method. Notice that this file is still in ASCII, but with redundancies removed. Unlike most compression methods, all the information present in the original files is still immediately accessible, without a decompression step.

As Figure 1 shows, using the compressed TASPI format saves considerable memory space. The amount of storage space saved is dependent on the amount of similarity between input trees. The more similarity between input trees (i.e. the greater the number of common subtrees) the more effective the compression. For the data sets shown, the compressed TASPI format uses just 5% of the storage requirement of the Newick format.

Figure 2 shows the time to compute consensus with each of TASPI, TNT and PAUP. In each case, both a strict consensus tree and a majority consensus tree are computed. Notice that the time to compute consensus includes the time to read the collection of trees since the trees are the input to a consensus calculation. Thus, we show both the time to compute consensus when reading compressed trees (TASPI.bhz line) and also the time when reading Newick trees (TASPI line).

In all cases, the result TASPI produces is identical to that produced by PAUP (when PAUP is able to read the input), but TASPI is faster. For the data sets PAUP and TNT can process that we present, using TASPI to compute consensus with input trees in compressed

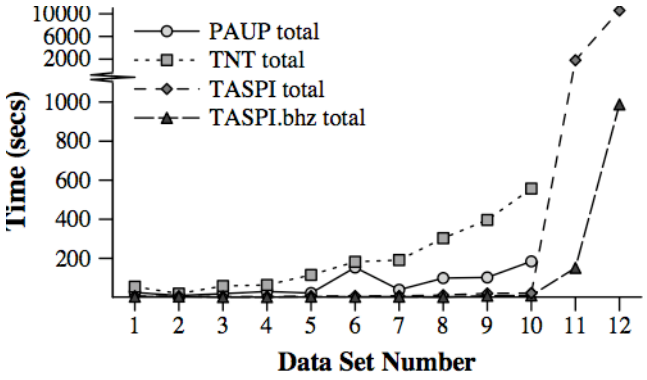

Figure 2: Time to read a collection of trees and compute their strict and majority consensus

TASPI format requires 6% of the time it takes PAUP to compute consensus with input trees in Newick format. If we factor out the improved reading time, TASPI computes these consensus trees in about 11% of the time it takes PAUP to do the same computation.

6. CONCLUSION

Using ACL2, we have begun a package for accurate phylogenetic tree manipulation. We have shown two representations of trees and proved that we can move accurately between them. Thus, in future work, we can encode other algorithms using trees as input in either format. We have also built a consensus method that gives (provably) well-formed results without losing performance, in fact, demonstrating decreased running times.

By programming in ACL2, we achieve confidence in our system. Verifying guards, and proving other key properties of the functions involved guarantees that TASPI does what it was designed to do. As we further this work, we hope to create a system that can reliably and efficiently answer questions biologists ask of both a particular phylogenetic tree, and a collection of trees.

While using ACL2 to implement further operations may not always give us the same dramatic performance improvement it did for consensus analysis, the ability to provide verified results is extremely useful.

Acknowledgments

This material is based upon work supported by the National Science Foundation under grant EF-0331453 and supported by DARPA and NSF under grant CNS-0429591. Also, Nelesen was supported by an NSF IGERT fellowship in Computational Phylogenetics and Applications to Biology at the University of Texas at Austin.

7. REFERENCES

[1] Nina Amenta, Frederick Clarke, and Katherine St. John. A linear-time majority tree algorithm. In Gary Benson and Roderic D. M. Page, editors, *Algorithms in Bioinformatics, Third International Workshop, WABI 2003, Budapest, Hungary, September 15-20, 2003, Proceedings*, volume 2812 of *Lecture Notes in Computer Science*, pages 216–227. Springer-Berlin / Heidelberg, 2003.

[2] Robert S. Boyer and Warren A. Hunt Jr. Function memoization and unique object representation for acl2 functions. In *Sixth International Workshop on the ACL2 Theorem Prover and its Applications (ACL2-2006)*, August 2006.

[3] Robert S. Boyer, Warren A. Hunt Jr., and Serita M. Nelesen. A compressed format for collections of phylogenetic trees and improved consensus performance. In *Algorithms in Bioinformatics: 5th International Workshop, WABI 2005*, number 3692 in Lecture Notes in Computer Science, pages 353–364. Springer-Verlag, 2005.

[4] B. Chor and T. Tuller. Finding the maximum likelihood tree is hard. In *Proceedings of the 9th Annual International Symposium on Research in Computational Biology (RECOMB 2005)*, 2005.

[5] W. Day, D. Johnson, and D. Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical Biosciences*, 81:33–42, 1986.

[6] William H. E. Day. Optimal Algorithms for Comparing Trees with Labeled Leaves. *Journal of Classification*, 2(1):7–28, 1985.

[7] J. Felsenstein. The Newick tree format. `http://evolution.genetics.washington.edu/phylip/newicktree.html`, 1986.

[8] Joseph Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Inc., 2004.

[9] P.A. Goloboff, J.S. Farris, and K.C. Nixon. TNT (Tree analysis using New Technology) (BETA) ver. 1.0. Published by the authors, Tucumán, Argentina, 2000.

[10] E. Goto, T. Soma, N. Inade, T. Ida, M. Idesawa, K. Hiraki, M. Suzuki, K. Shimizu, and B. Philpov. Design of a LISP Machine - FLATS. In *LFP '82: Proceedings of the 1982 ACM Symposium on LISP and functional programming*, pages 208–215, 1982.

[11] D. M. Hillis, J. P. Huelsenbeck, and C. W. Cunningham. Application and accuracy of molecular phylogenies. *Science*, 264:671–677, 1994.

[12] Matt Kaufmann, Pete Manolios, and J. S. Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, July 2000.

[13] David B. Searls. Pharmacophylogenomics: Genes, evolution and drug targets. *Nature Reviews Drug Discovery*, 2:613–623, 2003.

[14] D. L. Swofford. *PAUP*: Phylogenetic Analysis Using Parsimony (and Other Methods) 4.0 Beta*. Sinauer Associates, Sunderland, Massachusetts, 2002.