Tree Manipulation Language (TML)
Motivation

• Designing a language that simplifies programming trees
• Focuses more on tree operation rather than underlying data structures
• Provide simple operators for frequently used operations
• Precisely simple for programmers
Comparison

Insert into BST using C

/* insert a tnode into the binary tree */
struct tnode *tnode_insert(struct tnode *p, int value) {
    struct tnode *tmp_one = NULL;
    struct tnode *tmp_two = NULL;

    if (p == NULL) {
        /* insert [new] tnode as root node */
        p = (struct tnode *)malloc(sizeof(struct tnode));
        p->data = value;
        p->left = p->right = NULL;
    } else {
        tmp_one = p;
        /* Traverse the tree to get a pointer to the specific tnode */
        /* The child of this tnode will be the [new] tnode */
        while (tmp_one != NULL) {
            tmp_two = tmp_one;
            if (tmp_one->data > value)
                tmp_one = tmp_one->left;
            else
                tmp_one = tmp_one->right;
        }
    }
}

TML Style

As simple as alloc(a, b);
a->(b:^) or a->(^:b) depending on which child b is of a.
Language takes care of checking at compile time errors that you are thinking about
(Smart...huh.?)
tutorial to tml

treetype <2> MyTree_t
{
  int val = 0;
}
void main ()
{
  int i = 0;
  MyTree_t a, b, c, d, e, f;
  alloc(a, b, c, d, e, f);
  a -> (b -> (d : e -> (f : ~))): c;

  foreach node in a by preorder
  {
    node.val = i;
    i = i + 1;
  }

  foreach node in a by inorder
  {
    print (node.val);
    print (" ");
  }
  // other tree operators
  #ta;  // order among siblings
  ta[left];  // get child
  ta.val;  // get field
  &ta;  // get degree
  ^ta;  // get parent
  tb = @ta;  // copy the node only
  tc = $ta;  // copy the whole tree
}

inorder: 2 1 4 3 0 5
Implementation
Implementation

treetype <2, [left, right]> binary_tree
{
    int a;
    float b;
}

• Symbol table
  – parent
  – variables
  – functions
  – treetypes

  ▪ type name
  ▪ degree
  ▪ aliases
  ▪ members

162  Ent 0
163  Lfp -2
164  AIC 2
165  Fld i
166  Fld f
167  Pop 1
Lessons learnt

• SVN was a necessary to keep all member with a working version.
• Compromising when team had different opinions. Sometimes, vote to get it finalized.
• Designing was hard. But once it made sense, it benefits implementation.
• Testing suite kept record of what had done and what to do.
• Ask immediately via email if puzzled, rather than wait until meeting.
Conclusion

• Compiler
  – 586 ./analyzer.ml
  – 167 ./analyzer_test.ml
  – 56 ./ast.mli
  – 31 ./bytecode.mli
  – 427 ./generator.ml
  – 216 ./parser.mly
  – 50 ./sast.mli
  – 149 ./scanner.mll
  – 84 ./scanner_test.ml
  – 27 ./tml.ml
  – 45 ./type.mli
• 1838 subtotal

• Interpreter
  – 43 ./InorderIterator.java
  – 55 ./Instruction.java
  – 39 ./LevelorderIterator.java
  – 134 ./Main.java
  – 39 ./PostorderIterator.java
  – 39 ./PreorderIterator.java
  – 425 ./Program.java
  – 135 ./TMLTree.java
  – 10 ./Treeliterator.java
• 919 subtotal

• Total: 2757 lines of code
Conclusion

• Learning
• Contributing
• Enthusiastic
• More Learning
• More Contributing
• More Enthusiastic
• ....
• ....
• Enjoying