This is the central method of maketree.py. It takes in a list of TMRs and atomic actions

and attempts to place them into a hierarchical tree structure.

The function about\_part\_of(a,b) takes two TMRs and determines whether the second one has a THEME that HAS-OBJECT-AS-PART the THEME of the first one (that is, the first’s TMR is about an object that is part of the theme of the second’s TMR).

function construct\_tree:

make an empty TreeNode (that is, without a TMR or a name) to be the root

current\_parent = root (New nodes for non-atomic actions will be added  
 as children of whatever current\_parent is)

for each input (action or utterance) in the list of inputs:

if this input is an utterance, then

if there are no events in its tmr **then** continue

**else** if its tmr is in past tense: (that means its children came before it)

if about\_part\_of(current\_parent's tmr, new tmr) **then**

go up the tree from current\_parent until a node

is found for which this is not true, and set that

node as current\_parent

make a new TreeNode from the new tmr

set a pivot j such that, for each child c of current\_parent

**before** the jth, about\_part\_of(c, new) is false

set the children of current\_parent before the jth

as disputed between current\_parent and the new

node (that is, they might be children of either one,

and it is not known which)

put the children of current\_parent after the jth as

children of the new node, and not current\_parent

set the new node as a child of current\_parent

else if the previous input was an atomic action

(in the case where the about\_part\_of relation is false)

("previous node" refers to the most-recently-added child of current\_parent)

if that action currently is a child of a nameless node:

(that is, a non-primitive node without a tmr assigned)

let this tmr be the nameless node’s tmr

else if about\_part\_of(this new tmr, the tmr of the previous node):

current parent = previous node

insert a node with the new tmr between

current\_parent and its children

(that is, as a child of current\_parent and as the

parent of all of its children)

else:

mark all children of previous node

(which should all be atomic actions) as disputed

(i.e. they might belong to previous node,

but they might belong to this node)

else: (the about\_part\_of relation is false

and the previous input was not an atomic action)

this situation hasn't come up yet,

but more heuristics will be added here when it does

else: (the input's children come after it)

look up the tree to see if there is an ancestor of current\_parent such that   
 about\_part\_of(new tmr, ancestor's tmr) is true

if so, set that ancestor as current\_parent

make a new TreeNode with this tmr and set it as a child of current\_parent

if the next input is not an atomic action, set this new node as current\_parent,

so that the node for the next utterance will be added under the new node

else: (the input is an atomic action rather than an utterance)

if the last child of current\_parent already has children of its own:

make a child of current\_parent with no tmr (to be filled in with a tmr later)

set this action (and all consecutive atomic actions until the next utterance)

as children of the last child of current\_parent

disambiguate(root)

serialize the current state of the tree to be sent to the front-end

return root

SN: do you mean that until this moment the “new TMR” existed without being embedded in a node?

My mental picture of our current task is: all action elements of the input sequence denote primitive actions that will be leaves of the eventually learned hierarchy; all the utterances signal eventual non-terminals (so far it’s OK to check only whether the TMRs for these utterances are headed by events – down the road there will be more sophisticated heuristics here). So, whether we read in an action or an utterance, we should right away create a TreeNode for it – and then operate with this TreeNode: a) determining its parent, b) determining its children and c) naming it are the three main tasks. I think that you probably are already doing it this way – but not describing it this way for some reason. PW: Actually, there are some cases where a node without a TMR is created in the hierarchy, and then a TMR is assigned to it later, which is why I don’t immediately create a TreeNode.