Computations are easy
The structure of the abstract tree lends itself to the computations
that will need to be performed on it.
+ Note 3

Points assigned for this criteria: 100
Points earned with our syntax: 80

Easy to determine scope
The abstract tree is structured so that the built-in scoping rules
work on the tree in the same way that the language’s scoping rules
are intended to work.
+ Note 1
+ Note 4

Points assigned for this criteria: 60
Points earned with our syntax: 50+5 = 55

Minimum number of rules
The tree should be as simple as possible, while covering the entire
syntax.
− Note 2
+ Note 5
− Note 4

Points assigned for this criteria: 40
Points earned with our syntax: 25+5−1 = 29

Readability
The abstract tree should be as easily understandable as possible,
not only in rule and node naming, but also in structure where possible.
− Note 1
+ Note 2
− Note 4

Points assigned for this criteria: 40
Points earned with our syntax: 30−1 = 29

TOTAL:
(185+8)/240 = 80%

Notes:
1 Our syntax trees appear to differ from the language definition in the
case of 'let' statements. We parse a single let statement with multiple
definition lists as a tree of let statements each with a single
definition list. This may seem counterintuitive in parsing the language,
but greatly simplifies scoping.

2 We decided to develop our syntax with a bottom-up approach, first writing
a concrete syntax that can fully and correctly parse the language, and then
writing an abstract syntax that more cleanly represents the language without
leaving any automatically generated rules. This made our syntax a bit
redundantly defined.

3 We have already taken into account in designing the abstract syntax,
as much as possible, theory that has been touched on in class for the
type and name analysis computations that we will have to do in the future.

4 (Change 11/09/2005, +4 score)
We changed our tree to facilitate scope in function declaration and record
fields access and joined scopes in let expressions, and for loops. This
added some complexity to the tree, but makes scope determinations possible.

5 (Change 11/09/2005, +5 score)
We removed all of the redundant rules from the concrete syntax tree,
leaving only rules necessary to determine precedence and associativity,
and rules needed to solve conflicts.