Future of PCCTS

1. ANTLR
2. DLG
3. SORCERER
4. Implementation language of rewrites

1. ANTLR

Parser exception handling:

Convert

```
syntax error at "abc" missing { Operator "\)" }
```

to

```
if-statement: malformed conditional at "abc"
```

To “trap” any error occurring in rules invoked from the `if`-statement production or the production itself, we could use the following notation:

```
stat:   "if" expr "then" stat
        exception <<
            fprintf(stderr,
                    "if-statement: badly formed statement\n");
        >>
    ;
```

To catch only mismatched-token errors, we would indicate an exception type for the handler:

```
stat:   "if" expr "then" stat
        exception[MismatchedToken] <<
            fprintf(stderr,
                    "if-statement: mismatched token in statement\n");
        >>
    ;
```

To further restrict the handler to deal only with mismatched tokens:

```
stat:   "if" expr "then" stat
        exception.expr[MismatchedToken] <<
            fprintf(stderr, "if-statement: malformed condition\n");
        >>
    ;
```

- Ability to inherit rules from other grammars (in C++ mode).
- Change of input format:
  - `<<...>>` → {...}
  - {...} (optional construct) → `[..]`
  - `rule[args]` → `rule<args>`
- Labels (like SORCERER) instead of "`$i`".
- Add lexical rules to ANTLR:

```
ID : ( 'a'..'z' )+ ;
```

- Total rewrite: fix input format, reduce feature explosion, better designed C and C++ output.

2. DLG

- Add backtracking.
- "Deep six" it?

3. SORCERER

- More tree rewrite routines.
- Integrated symbol management?
- Interface (that would translate to "raw" SORCERER) to handle common operations.
- C++ output.

4. Implementation language for rewrites

Requirements:

1. Efficient and portable (remember the exponential nature of grammar analysis).
2. Modifiable by PCCTS users.
3. Extremely widespread or SMALL public-domain implementation and library.

Possibilities:

- C
- C++
- Objective-C
- Sather
- Smalltalk
- TOOL: Ter's Object-Oriented Language?