Theoretical Results Omitted from "LL(*): *The Foundation of the ANTLR Parser Generator*"

Authors omitted for blind review

November 19, 2010

1 Theoretical Results

Theorem 1 *Every PEG without syntactic predicates has an equivalent ANTLR grammar.*

Proof. *After converting meta-language syntax, it is sufficient to restrict lookahead to $k = 1$ in every decision and prefix every alternative in the ANTLR grammar, α_i, with syntactic predicate* `(`α_i`)=>`. *For every decision, there are two cases to consider. Case (i): The decision is $LL(1)$, which means each input symbol $b \in 1 : LA_i$ predicts a single alternative i. A PEG tries the alternatives in order until it reaches alternative i. ANTLR's lookahead DFA has edge $D_0 \xrightarrow{b} f_i$ and immediately jumps to the same alternative without speculating.*

Case (ii): The decision is non-$LL(1)$. A PEG will pick the first alternative that matches at the current input position. ANTLR's grammar analysis creates a DFA with an edge for every $b \in 1 : LA_i$, $D_0 \xrightarrow{b} p_b$. p_b is accept state f_i when b uniquely predicts alternative i. When b predicts multiple alternatives, ANTLR adds DFA edges $p_b \xrightarrow{synpred(\alpha_i)} f_i$ for all alternatives i nondeterministic upon b in the order specified in the grammar. The DFA tests the predicated edges in that same order. The DFA effectively restricts backtracking to only those alternatives that begin with b. Order is preserved and so the ANTLR grammar matches the same alternative as the PEG, albeit without testing alternatives not starting with b. ***TODO:*** *add sem preds?*

For example, the following ANTLR grammar has the same hazard as the equivalent PEG, $A \leftarrow a/ab$.

```
a options {k=1;}
    :  (A)=>   A
    |  (A B)=> A B
    ;
```

Input token `A` predicts both alternatives and so ANTLR will try them in order, erroneously matching the first alternative upon input `A B`.

Theorem 2 *Every PEG with positive and negative syntactic predicates can be transformed to an equivalent to ANTLR grammar.*

Proof. *By construction. First we convert the PEG to an ANTLR grammar per the previous proof. Then, we erase the syntactic predicates to ANTLR semantic predicates. PEG syntactic predicate* `&`E *becomes ANTLR semantic predicate* `{synpred("predE")}?` *and we introduce a rule for the CFG fragment E,* `predE : `E `;`. *Similarly, we translate PEG "not predicate"* `!`E *to semantic predicate* `{!synpred("predE")}?`. *Function* `synpred` *invokes the specified rule and returns true if the rule matches from the current input pointer and returns false otherwise. The state of the parser is preserved after the call.*

For example, Ford [1] provides the following PEG for non-context-free language $a^n b^n c^n$:

$A \leftarrow aAb \,/\, \epsilon$
$B \leftarrow bBc \,/\, \epsilon$
$D \leftarrow \&(A!b)\, a^* B\, !.$

Mechanically following the predicate erasure rules, we get the following ANTLR grammar (where nonterminals start with lowercase letters, tokens with uppercase):

```
options { k=1; backtrack=true; }
a : A a B | ;
b : B b C | ;
d : {synpred("pred1")}? A* b EOF ;
pred1 : a {!synpred("pred2")}? ; // a then not B
pred2 : B ;
```

The availability of unrestricted semantic predicates in ANTLR supports context-sensitive parsing in a practical but *ad hoc* manner.

LL(*) speculates less often than packrat parsing unless every alternative starts with the same input symbol. Without the $k = 1$ restriction, *LL(*)* has the potential to speculate even less. Theory agrees with practice as we show in the next section.

References

[1] Ford, B. Parsing expression grammars: a recognition-based syntactic foundation. In *POPL'04* (2004), ACM Press, pp. 111–122.