Putting the Pieces together

predict sets for rules

- Remember: a recursive descent parser has one function for each non-terminal
- How do we decide which non-terminal to match for a rule?
- Build a **predict set** for each rule: the set of terminals we would want to see to predict rewriting the non-terminal with this rule

$S \rightarrow X\ Y\ \$$

$X \rightarrow a\ Y\ q$

$X \rightarrow b$

$X \rightarrow Yq$

$Y \rightarrow \lambda$

$Y \rightarrow d$

predict sets for rules

- Predict($X \rightarrow \alpha$) =

  First(α) if $\lambda \notin$ First(α)
  (If the right-hand side cannot become the empty string, the terminals that this rule can generate come from the first set of the RHS)

  (First(α) - λ) U Follow(X) otherwise
  (If the RHS can become the empty string, then this rule can be used to "throw away" X, so need to consider what might come after X)

S → X Y $
X → a Y q
X → b
X → Yq
Y → λ
Y → d

predict sets for rules

- $\text{Predict}(X \rightarrow \alpha)$ =

  $\text{First}(\alpha)$ if $\lambda \notin \text{First}(\alpha)$
  (If the right-hand side cannot become the empty string, the terminals that this rule can generate come from the first set of the RHS)

  $(\text{First}(\alpha) - \lambda)$ U $\text{Follow}(X)$ otherwise
  (If the RHS can become the empty string, then this rule can be used to "throw away" X, so need to consider what might come after X)

S → X Y $

X → a Y q

X → b

X → Yq

Y → λ

Y → d

First(S) = {a, b, d, q}

First(X) = {a, b, d, q}

First(Y) = {d, λ}

Follow(S) = { }

Follow(X) = {d, $}

Follow(Y) = {q, $}

predict sets for rules

- Predict($X \rightarrow \alpha$) =

  First(α) if $\lambda \notin$ First(α)
  (If the right-hand side cannot become the empty string, the terminals that this rule can generate come from the first set of the RHS)

  (First(α) - λ) U Follow(X) otherwise
  (If the RHS can become the empty string, then this rule can be used to "throw away" X, so need to consider what might come after X)

S → X Y $

X → a Y q

X → b $\{\boldsymbol{b}\}$

X → Yq

Y → λ

Y → d

First(S) = {a, b, d, q}
First(X) = {a, b, d, q}
First(Y) = {d, λ}

Follow(S) = { }
Follow(X) = {d, $}
Follow(Y) = {q, $}

predict sets for rules

- Predict($X \rightarrow \alpha$) =

  First(α) if $\lambda \notin$ First(α)
  (If the right-hand side cannot become the empty string, the terminals that this rule can generate come from the first set of the RHS)

  (First(α) - λ) U Follow(X) otherwise
  (If the RHS can become the empty string, then this rule can be used to "throw away" X, so need to consider what might come after X)

S → X Y $

X → a Y q

X → b $\{b\}$

X → Yq

Y → λ

Y → d $\{d\}$

First(S) = {a, b, d, q}

First(X) = {a, b, d, q}

First(Y) = {d, λ}

Follow(S) = { }

Follow(X) = {d, $}

Follow(Y) = {q, $}

predict sets for rules

- Predict($X \rightarrow \alpha$) =

  First(α) if $\lambda \notin$First(α)
  (If the right-hand side cannot become the empty string, the terminals that this rule can generate come from the first set of the RHS)

  (First(α) - λ) U Follow(X) otherwise
  (If the RHS can become the empty string, then this rule can be used to "throw away" X, so need to consider what might come after X)

S → X Y $	
X → a Y q	$\{a\}$
X → b	$\{b\}$
X → Yq	
Y → λ	
Y → d	$\{d\}$

First(S) = {a, b, d, q}
First(X) = {a, b, d, q}
First(Y) = {d, λ}

Follow(S) = { }
Follow(X) = {d, $}
Follow(Y) = {q, $}

predict sets for rules

- Predict($X \rightarrow \alpha$) =

  First(α) if $\lambda \notin$ First(α)
  (If the right-hand side cannot become the empty string, the terminals that this rule can generate come from the first set of the RHS)

  (First(α) - λ) U Follow(X) otherwise
  (If the RHS can become the empty string, then this rule can be used to "throw away" X, so need to consider what might come after X)

S → X Y $

X → a Y q $\{a\}$

X → b $\{b\}$

X → Yq

Y → λ $\{q, \$\}$

Y → d $\{d\}$

First(S) = {a, b, d, q}
First(X) = {a, b, d, q}
First(Y) = {d, λ}

Follow(S) = { }
Follow(X) = {d, $}
Follow(Y) = {q, $}

predict sets for rules

- Predict($X \to \alpha$) =

  First(α) if $\lambda \notin$First(α)
  (If the right-hand side cannot become the empty string, the terminals that this rule can generate come from the first set of the RHS)

  (First(α) - λ) U Follow(X) otherwise
  (If the RHS can become the empty string, then this rule can be used to "throw away" X, so need to consider what might come after X)

Rule	Predict
S → X Y $	
X → a Y q	$\{a\}$
X → b	$\{b\}$
X → Yq	$\{d, q\}$
Y → λ	$\{q, \$\}$
Y → d	$\{d\}$

First(S) = {a, b, d, q}
First(X) = {a, b, d, q}
First(Y) = {d, λ}

Follow(S) = { }
Follow(X) = {d, $}
Follow(Y) = {q, $}

predict sets for rules

- Predict($X \to \alpha$) =

  First(α) if $\lambda \notin$First(α)
  (If the right-hand side cannot become the empty string, the terminals that this rule can generate come from the first set of the RHS)

  (First(α) - λ) U Follow(X) otherwise
  (If the RHS can become the empty string, then this rule can be used to "throw away" X, so need to consider what might come after X)

S → X Y $	$\{a, b, d, q\}$
X → a Y q	$\{a\}$
X → b	$\{b\}$
X → Yq	$\{d, q\}$
Y → λ	$\{q, \$\}$
Y → d	$\{d\}$

First(S) = {a, b, d, q}
First(X) = {a, b, d, q}
First(Y) = {d, λ}

Follow(S) = { }
Follow(X) = {d, $}
Follow(Y) = {q, $}

building the parser

- Build the function for each non-terminal:
  - Switch on the lookahead token in the string, pick rule to expand based on predict sets of the rules
  - Match the rule:
    - If a terminal, match against the string
    - If a non-terminal, invoke that non-terminal’s function

next: does this always work?