As we have seen, in order to make shift-reduce parsing practical, we need a reasonable way to identify viable prefixes (and so, possible handles).

Up to now, it has not been clear that this can be done without considering the whole stack contents (to decide whether the string on the stack is a prefix of at least one right-sentential form whose rightmost handle extends to at least the end of the stack contents).

Remarkably, in LR parsers this is nicely done, using a DFA to recognize viable prefixes!

We’ll consider approaches to LR parsing, of varying complexity and generality, beginning with the simplest — SLR parsing.

(SLR — S “simple” L “left-to-right scanning” R “rightmost derivation”)

**LR(0) items**

**Definition** An LR(0) item, or simply an item, of a grammar $G$ is a production of $G$ with a single dot inserted at some position in the rhs.

For example, the production

$$S \to ABC$$

yields four items

$$S \to \cdot ABC \mid A \cdot BC \mid AB \cdot C \mid ABC \cdot$$

The production

$$A \to \epsilon$$

yields a single item

$$A \to \cdot$$

Intuitively, an item indicates how much of the rhs of a production we have accounted for at a given point in a parse. For instance, the item

$$S \to A \cdot BC$$

indicates that we have seen a string derivable from $A$ and hope to see a string derivable from $BC$. 
When an item is valid for a viable prefix

**Definition** An item \( A \rightarrow \beta_1 \cdot \beta_2 \) is valid for a viable prefix \( \alpha \beta_1 \) if there is a \( \gamma \) s.t. \( \alpha A \gamma \) is a right-sentential form from which there is a rightmost derivation of \( \alpha \beta_1 \beta_2 \gamma \).

**Observation** It follows that \( A \rightarrow \beta_1 \cdot \beta_2 \) is valid for a viable prefix \( \alpha \beta_1 \) iff there is a \( \gamma \) s.t. \( A \rightarrow \beta_1 \beta_2 \) is handle for \( \alpha \beta_1 \beta_2 \gamma \) in the position following \( \alpha \).

Intuitively, this means that the production corresponding to the item may turn out to be a handle.

**Example**

\[
S \rightarrow cAd \\
A \rightarrow ab \mid a \\
S \rightarrow \cdot cAd \mid c \cdot Ad \mid cA \cdot d \mid cAd \\
A \rightarrow \cdot ab \mid a \cdot b \mid ab \cdot \mid \cdot a \mid a \\
\]

\( S \rightarrow \cdot cAd \) is valid for \( \epsilon \), since \( S \Rightarrow cAd \).

\( S \rightarrow c \cdot Ad \) is valid for \( c \), since \( S \Rightarrow cAd \).

\( A \rightarrow \cdot ab \) is valid for \( c \), since \( cAd \Rightarrow cabd \).

\( A \rightarrow \cdot a \) is valid for \( c \), since \( cAd \Rightarrow cad \).

An item \( A \rightarrow \beta_1 \cdot \beta_2 \) is valid for a viable prefix \( \alpha \beta_1 \) if there is a \( \gamma \) s.t. \( \alpha A \gamma \) is a right-sentential form from which there is a rightmost derivation of \( \alpha \beta_1 \beta_2 \gamma \).

\[
S \rightarrow \cdot cAd \mid c \cdot Ad \mid cA \cdot d \mid cAd \\
A \rightarrow \cdot ab \mid a \cdot b \mid ab \cdot \mid \cdot a \mid a \\
\]

\( A \rightarrow a \cdot b \) is valid for \( ca \), since \( cAd \Rightarrow cabd \).

\( A \rightarrow a \cdot d \) is valid for \( ca \), since \( cAd \Rightarrow cad \).

Is \( S \rightarrow cA \cdot d \) valid for \( ca \)?

\( A \rightarrow ab \cdot d \) is valid for \( cab \), since \( cAd \Rightarrow cabd \).

\( S \rightarrow cA \cdot d \) is valid for \( cA \), since \( S \Rightarrow cAd \).

\( S \rightarrow cAd \cdot d \) is valid for \( cAd \), since \( S \Rightarrow cAd \).

Roughly, if an item whose rhs is dotted at the far right is valid for the stack contents, we can reduce.

(Why? There is something we could add to the stack contents to obtain a right-sentential form for which the valid item’s production is a handle at the top of the stack. Note that, in this formulation, we do not take into account any lookahead.)

So we want a nice way to identify valid items during the course of a parse...
$S \rightarrow cAd$ is valid for $\epsilon$, since $S \Rightarrow cAd$.

Initial stack contents are viable; valid item corresponds to lone $S$-production. (At start, valid items include left-dotted versions of the $S$-productions. And, if any of those items have a variable $X$ after the dot, then the valid items include the left-dotted versions of the $X$-productions, and so on.)

$S \rightarrow c \cdot A d$ is valid for $c$, since $S \Rightarrow cAd$.

$A \rightarrow a \cdot b$ is valid for $ca$, since $cAd \Rightarrow cabd$.

$A \rightarrow a \cdot$ is valid for $ca$, since $cAd \Rightarrow cad$.

$A \rightarrow a$ is valid for $ca$, since $cAd \Rightarrow cad$.

$c$ is a viable prefix, with 3 valid items, the first of which corresponds to the prior valid item. The other 2 valid items represent, roughly speaking, the available ways to get an $A$ onto the stack after the $c$, which a successful parse in the grammar must do. That is, we’ll have to reduce using an $A$-production in order to move right (over the $A$) in the 1st item.

$A \rightarrow a \cdot b$ is valid for $ca$, since $cAd \Rightarrow cabd$.

$cA$ is viable, with valid item corresponding to one of the valid items for $c$.

$S \rightarrow cA \cdot d$ is valid for $cA$, since $S \Rightarrow cAd$.

$cAd$ is viable, with one valid item corresponding to the unique valid item for $cA$.

$S$ is viable, but with no valid item. (The start symbol is always viable.)
An NFA that accepts the viable prefixes and identifies the valid items for each viable prefix

Given a grammar with start symbol S, consider the augmented grammar with new start symbol $S'$ and additional production $S' \rightarrow S$.

We construct an NFA as follows:

- The states are the items of the augmented grammar.
- The initial state is $S' \rightarrow \cdot S$.
- All states are accepting.
- For any pair of items of the form $A \rightarrow \alpha \cdot X \beta$ and $A \rightarrow \alpha X \cdot \beta$ there is a transition from the first to the second labeled $X$.
  (Here $X$ can be either a variable or a terminal.)
- For any pair of items of the form $A \rightarrow \alpha \cdot B \beta$ and $B \rightarrow \cdot \gamma$ there is an $\epsilon$-transition from the first to the second.
  (Notice that $B$ is a variable.)

The states are the items of the augmented grammar.

For any two items of the forms $A \rightarrow \alpha \cdot X \beta$ and $A \rightarrow \alpha X \cdot \beta$ there is a transition from the first to the second labeled $X$.

For any two items of the forms $A \rightarrow \alpha \cdot B \beta$ and $B \rightarrow \cdot \gamma$ there is an $\epsilon$-transition from the first to the second.

All states are accepting. The initial state is $S' \rightarrow \cdot S$.

Claim 1 This NFA accepts the language of all viable prefixes (of the original grammar)!

Claim 2 The states of the NFA that you can reach by reading a string $\alpha$ are exactly the items (of the augmented grammar) that are valid for $\alpha$.

Example

$S' \rightarrow \cdot S | S$

$S \rightarrow \cdot cAd | c \cdot Ad | cA \cdot d | cAd$

$A \rightarrow \cdot ab | a \cdot b | ab \cdot | \cdot a | a$

What states can the NFA reach by reading $\epsilon$? (What is $\epsilon$-closure($\{S' \rightarrow \cdot S\}$)?)

$\{S' \rightarrow \cdot S, S \rightarrow \cdot cAd\}$

What states can the NFA reach by reading $c$?

$\{S \rightarrow \cdot cAd, A \rightarrow \cdot ab, A \rightarrow \cdot a\}$

What states can the NFA reach by reading $ca$?

$\{A \rightarrow a \cdot b, A \rightarrow a\}$
The states are the items of the augmented grammar.

For any two items of the forms $A \to \alpha \cdot X \beta$ and $A \to \alpha X \cdot \beta$
there is a transition from the first to the second labeled $X$.

For any two items of the forms $A \to \alpha \cdot B \beta$ and $B \to \cdot \gamma$
there is an $\epsilon$-transition from the first to the second.

All states are accepting. The initial state is $S' \to \cdot S$.

$S' \to \cdot S \mid S$

$S \to \cdot cAd \mid c \cdot Ad \mid cA \cdot d \mid cAd \cdot$

$A \to \cdot ab \mid a \cdot b \mid ab \cdot \mid \cdot a \mid a \cdot$

What states can the NFA reach by reading $cab$?
Recall that $ca$ takes the NFA to $\{ A \to a \cdot b, A \to a \cdot \}$. So $cab$ goes to $\{ A \to ab \cdot \}$.

What states can the NFA reach by reading $cA$?
Recall that $c$ takes the NFA to $\{ S \to c \cdot Ad, A \to \cdot ab, A \to \cdot a \}$

What states can the NFA reach by reading $cAd$?

What states can the NFA reach by reading $a$?
Recall: $\epsilon$-closure($\{ S' \to \cdot S \}$) = $\{ S' \to \cdot S, S \to \cdot cAd \}$.

Neither of those states has an outgoing $a$-transition.

So we can construct an NFA that

- accepts only viable prefixes
- identifies valid items for each viable prefix

In the standard way, we can reduce this NFA to a DFA.

**Example**

$S' \to \cdot S \mid S$

$S \to \cdot cAd \mid c \cdot Ad \mid cA \cdot d \mid cAd \cdot$

$A \to \cdot ab \mid a \cdot b \mid ab \cdot \mid \cdot a \mid a \cdot$

In the DFA, the states are:

0. $\{ S' \to \cdot S, S \to \cdot cAd \}$
1. $\{ S \to c \cdot Ad, A \to \cdot ab, A \to \cdot a \}$
2. $\{ A \to a \cdot b, A \to a \cdot \}$
3. $\{ A \to ab \cdot \}$
4. $\{ S \to cA \cdot d \}$
5. $\{ S \to cAd \cdot \}$
6. $\{ S' \to \cdot S \}$

We will use this DFA to guide the actions of a shift-reduce parser.
For next time

Continue reading 4.7.