LL(k) Parser Construction

OTIVATION:

Nice to have; needed to strengthen parsers

ROBLEM:

Exponentially complex merely to store lookahead $O(|T|^k)$

OLUTION

Use as little lookahead as possible

Use approximate *LL(k)* when possible $O(|T| \times k)$

Use combined Approximate/Full *LL(k)* as last resort

Use sequence of prediction expressions rather than one giant expression

pproximate Lookahead: $LL_1(2)$

```
:   (A B|C D|F E|H K|J K|A M)
|   F I
;
```

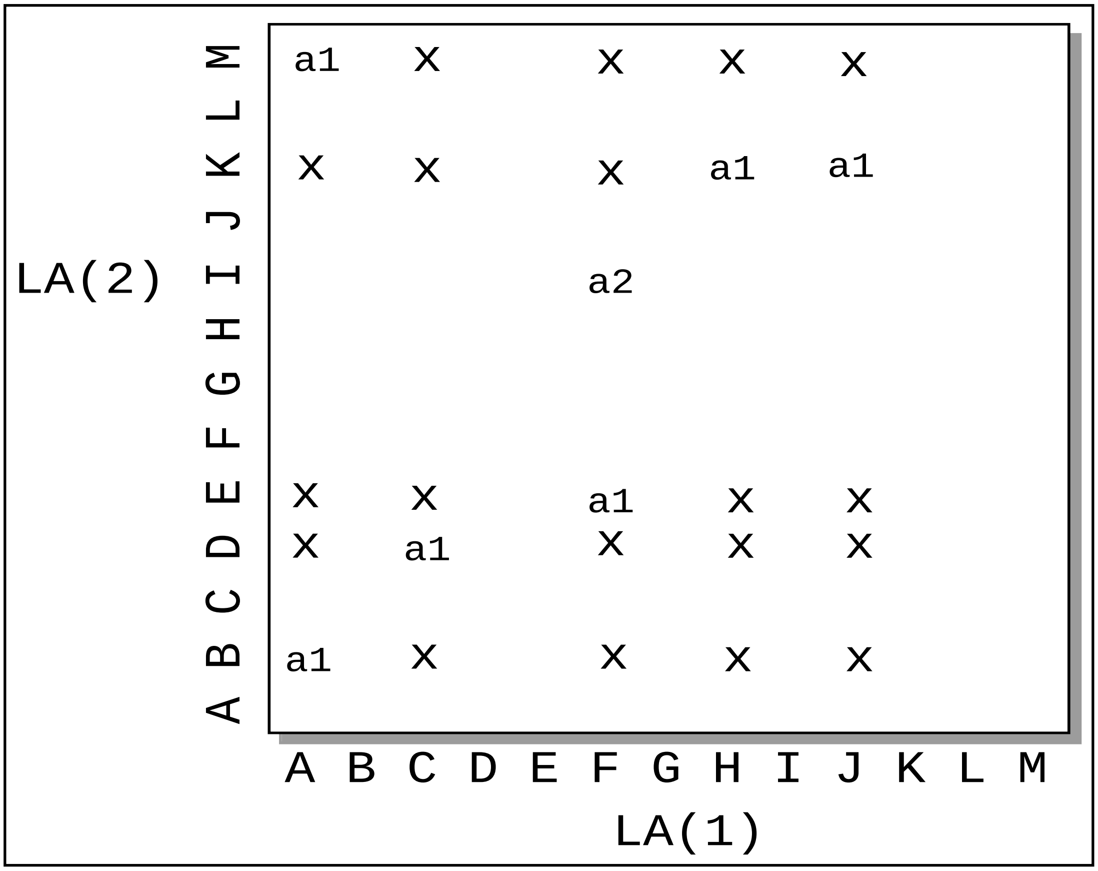
LA(2)

	A	B	C	D	E	F	G	H	I	J	K	L	M
M	a1												
L													
K								a1		a1			
J													
I						a2							
H													
G													
F													
E						a1							
D			a1										
C													
B	a1												
A													

LA(1)

pproximate Lookahead is Cross Product of

LA(1) x LA(2)



LA(2)

	A	B	C	D	E	F	G	H	I	J	K	L	M
M	a1		X			X		X		X			
L													
K	X		X			X		a1		a1			
J													
I						a2							
H													
G													
F													
E	X		X			a1		X		X			
D	X		a1			X		X		X			
C													
B	a1		X			X		X		X			
A													

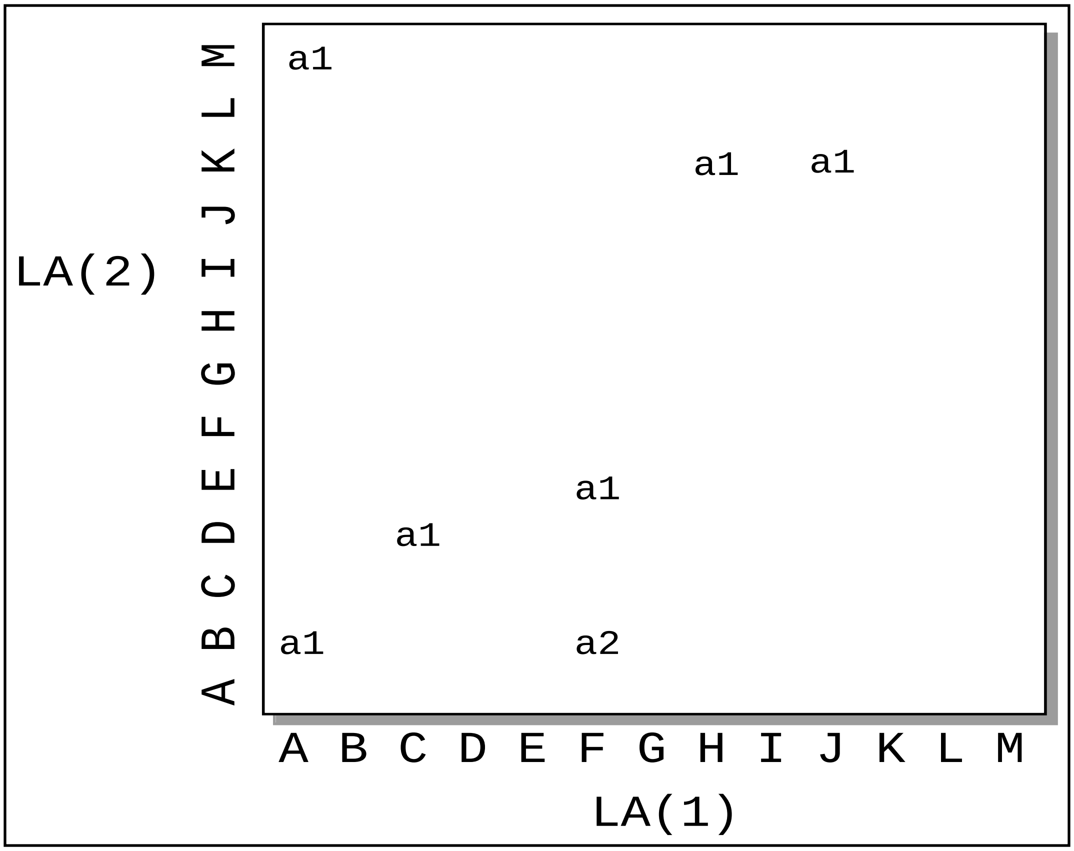
LA(1)

```
* Approximate lookahead decisions */
()

  if (  LA(1) in {A,C,F,H,J} &&
        LA(2) in {B,D,E,K,M} )
  {
     // match alternative 1
  }
  else if ( LA(1)==F && LA(2)==I )
  {
     // match alternative 2
  }
  else syntax_error;
```

ormal (full) Lookahead: *LL*(2)

```
:   (A B|C D|F E|H K|J K|A M)
|   F B
;
```



LA(2) \ LA(1)	A	B	C	D	E	F	G	H	I	J	K	L	M
M	a1												
L													
K								a1		a1			
J													
I													
H													
G													
F													
E						a1							
D			a1										
C													
B	a1					a2							
A													

ross Product Collides With Sequence From Alternative 2

LA(2)	A	B	C	D	E	F	G	H	I	J	K	L	M
M	a1		X			X		X		X			
L													
K	X		X			X		a1		a1			
J													
I													
H													
G													
F													
E	X		X			a1		X		X			
D	X		a1			X		X		X			
C													
B	a1		X			X a2		X		X			
A													

LA(1)

herefore, approximate lookahead is nondeterministic ull Lookahead Achieved With Help of Approximate:

Do not resort to 2-tuple comparison

Simply test for collisions of cross-product with tuples from other alternatives

```
 Full lookahead decisions */
)

  if (    LA(1) in {A,C,F,H,J} &&
          LA(2) in {B,D,E,K,M} &&
          !(LA(1)==F&&LA(2)==B) )
  {
      // match alternative 1
  }
  else if ( LA(1)==F && LA(2)==B )
  {
      // match alternative 2
  }
  else syntax_error;
```

rder of Alternative Evaluation Important

```
  :   F B
  |   (A B|C D|F E|H K|J K|A M)
  ;

 Full lookahead decisions */
)

  if ( LA(1)==F && LA(2)==B )
  {
      // match alternative 1
  }
  else if (   LA(1) in {A,C,F,H,J} &&
              LA(2) in {B,D,E,K,M} )
  {
      // match alternative 2
  }
  else syntax_error;
```

o longer need to test for collision with cross product

Controlling use of approximate lookahead

ssibilities:

-ck n command line option. Sets maximum approximate ookahead before full LL(k) is attempted.

#pragma approx for use on a per subrule basis. Indicates hat the modified subrule should be handled with approximate lookahead—full LL(k) is either unnecessary or makes ANTLR go off to lunch and never come back.

amples...