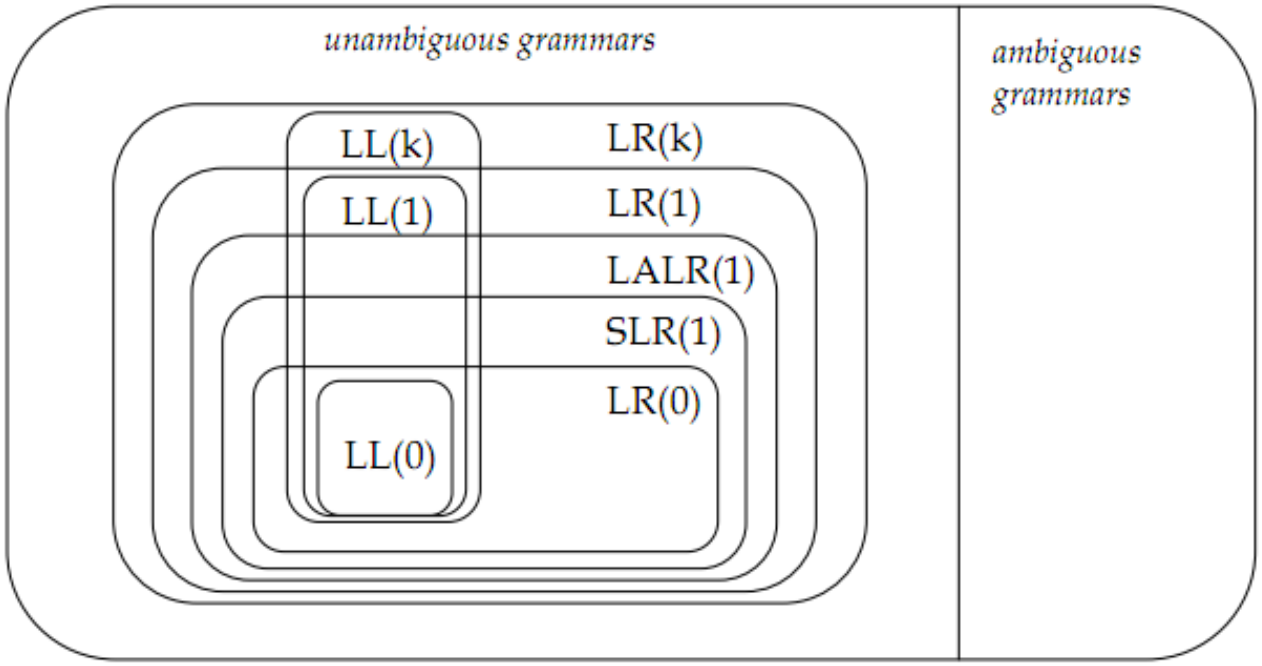
Why we have many parsers?



What we want?

- Correctness (soundness) – a MUST
  - When input is in the L, say “yes”. Otherwise say “no”.
- Completeness
  - Ability to recognize any CFL
  - Not a must. We can compromise on this.
- Efficient
  - Time wise
  - Space wise
  - We can go for a better resource usage method. We can compromise on this. {at the expense of resource wastage}.

Not LR(0),..?

- What do you mean by saying, a CFG is not LR(0)? Or, not SLR? …
- The parse table is going to have multiple entries for some (state, input) pairs.
  - With a choice being made, we may say that the string is not in the language (eventhough it is in the language)
  - With correct choice we say, the string is in the language.
- A weaker one can say “no” (even when the answer is yes).
  - So, even before doing parsing, you can say that your correctness is compromised.
- For a class of CFGs you can be correct.
  - This is what is LR(0) class.

We can be correct even with weak ones.

- If backtracking is allowed, weakest parser also has same power as a stronger one.
  - Before saying “no”, verify whether any options are not explored …
- But, this makes the time to parse more than linear (in the input string size).
  - Parsing can become extremely slow.

- Let us say LR(0) says that the string is there in the language.
  - Then SLR also says so.
  - CLR also says so.
- In fact, the shift or reduce actions followed by each of these parsers are exactly same.

Let the CFG is LR(0) and the string is not in the language

- Then LR(0) will say “no”.
- SLR and CLR also will say same.

- Where is the difference?

Where is the difference?

- CLR can say so quickly, than SLR
- SLR can say so quickly, than LR(0)

- Error recognition capability is different for different parsers.
- This affects the time!
  - Time is reduced.
  - What about space?

- So the stronger the parser is,
  - No conflicts, ..
  - More blanks (indicating errors)
  - The class of CFGs recognized is a larger one.

A notable fact

- Filling blanks with reductions (say, r3) doesn't affect the correctness.
  - But can increase time.
- You cannot fill blanks with shifts
  - This alters the automaton.
  - The language recognized is altered.
  - For the same reason, you cannot play with GOTO.