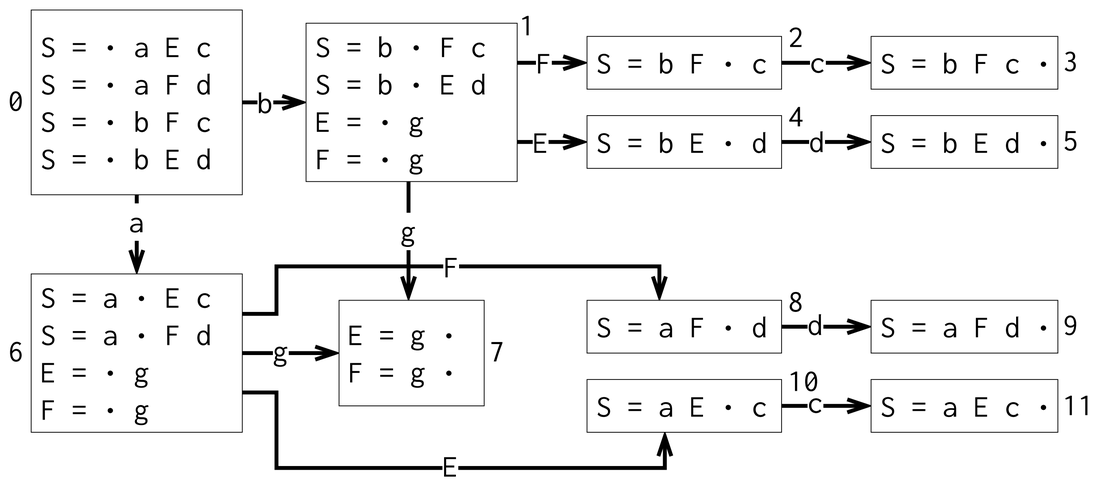
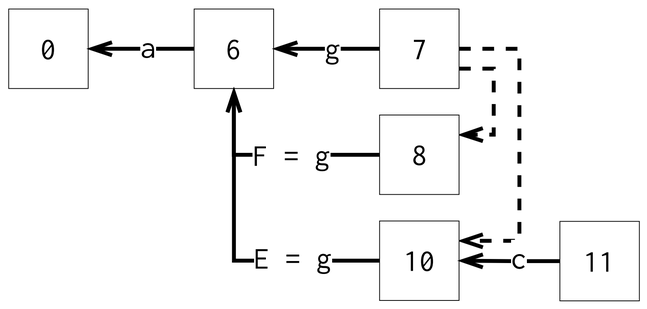
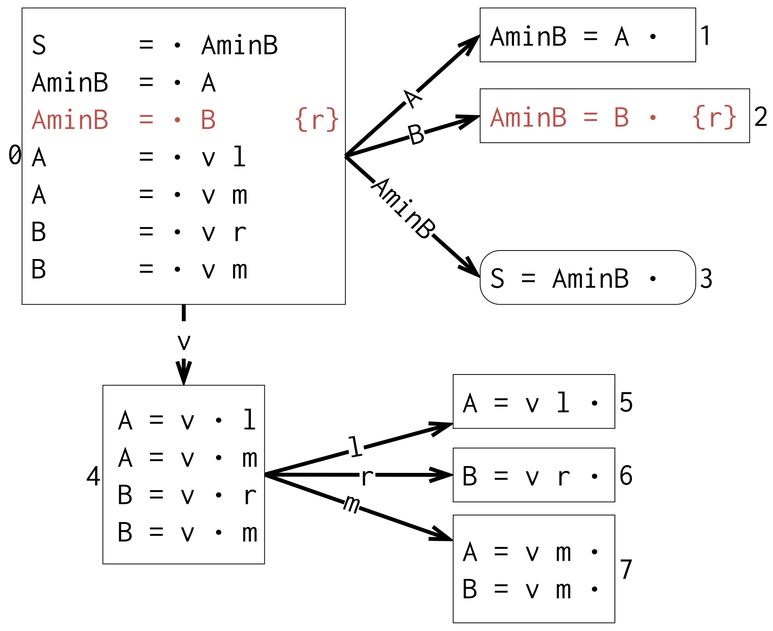
Ordering Rejectable Stacks in SGLR Parsing





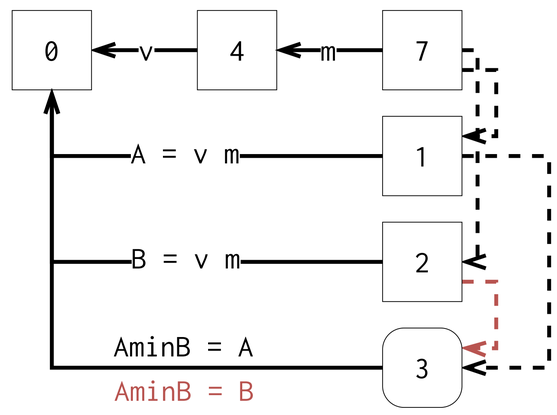
Generalised LR (GLR) Parsing

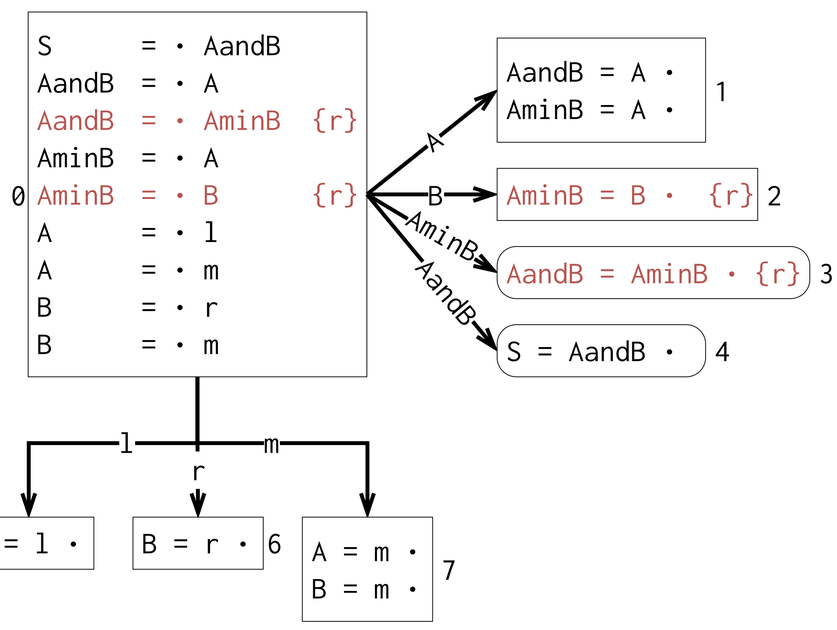
Use an LR parsing automaton, at a conflict do all options in parallel by using multiple stacks. The stacks are shared at each level (level = no. of input tokens seen). This creates a graph, the Graph Structured Stack (GSS).

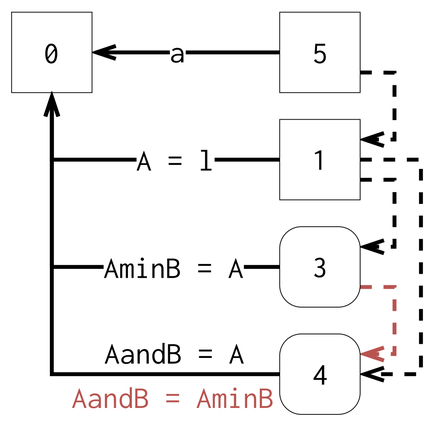


Scannerless GLR (SGLR) Parsing

Without a scanner, the input tokens are individual characters. This makes entire grammars closed under composition. It does cause more conflicts in the parser. Resolving ambiguities like keyword-over-identifier is done with **reject rules**. These exclusion rules are one of the alternatives for a non-terminal and eliminate the successful parse of any other alternatives for that non-terminal if it matches.

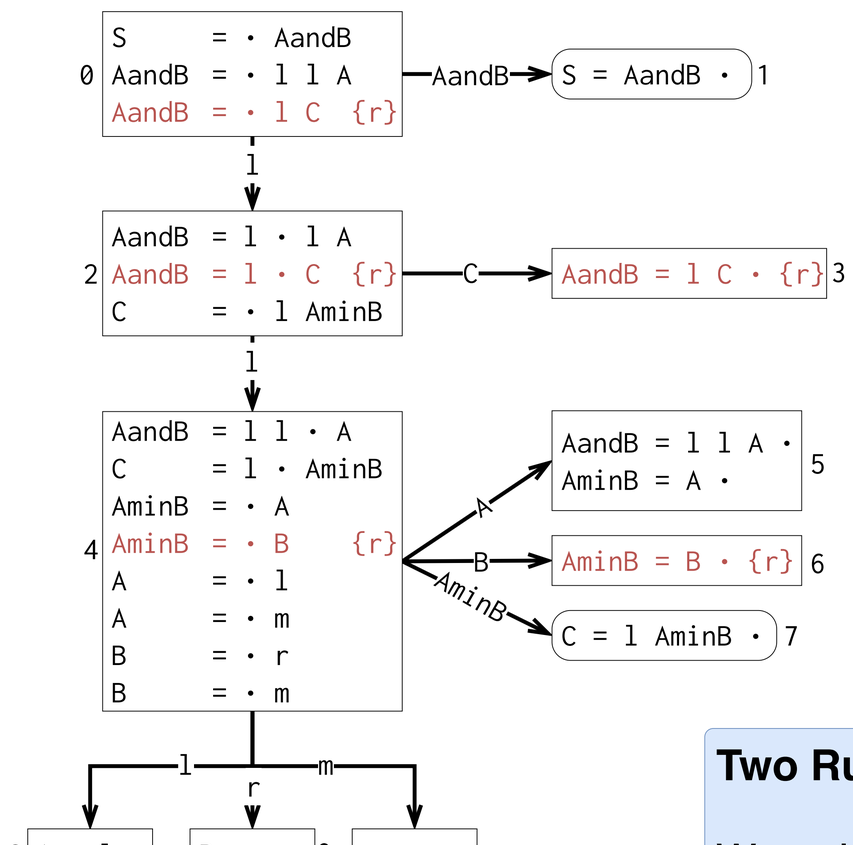






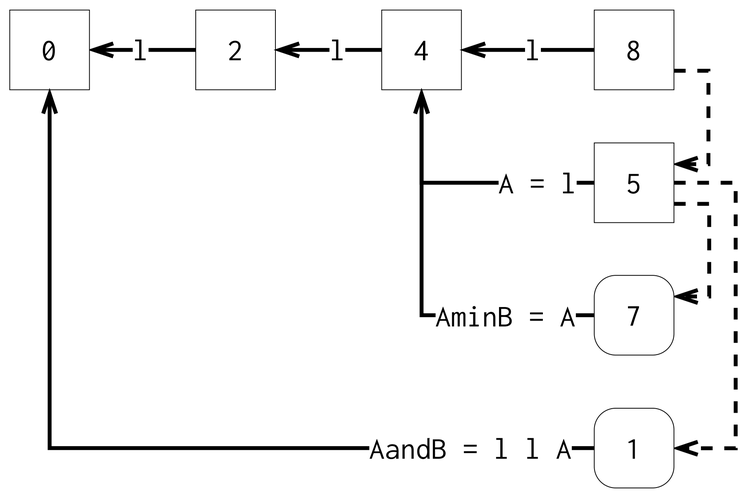
Ordering Rejectable States?

Filtering a highly ambiguous parse forest after parsing is a very expensive way to implement reject rules correctly. It is much cheaper to do during parsing. But the published works on SGLR for parse-time filtering are incomplete: First reduce on stacks with normal states on top, then on rejectable stacks, i.e. with rejectable states on top (rounded corners). This ordering is not enough if multiple rejectable stacks can influence each other. The example above parses correctly with this method, but the example to the left needs an additional ordering between states 3 and 4.



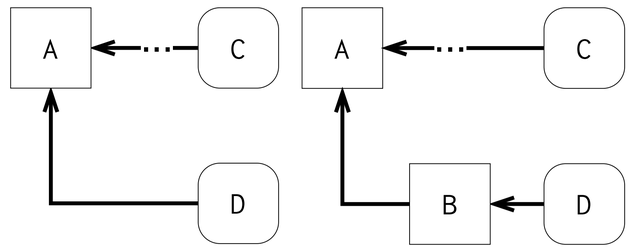
Ordering Rejectable Stacks!

While ordering between states looks fairly local in the last example by using their shared "parent" state, a minor change to the grammar puts the rejectable states much further away in the automaton. But note how the rejectable stacks now look quite different. Only 7 can reject 1, not the other way around, as rejects eventually shorten a stack and contribute the reject reduction to the link from the shared ancestor.



Two Rules and a Pre-Computed Ordering

We only need two rules to order rejectable stacks. One is the "longer stack first" principle above, the other is the state ordering for equal length stacks, using the parent state information. This can be pre-computed based on the automaton. In the 2-page abstract you can find a more details.



Jeff Smits, Daniel A. A. Pelsmaeker

TU Delft Delft University of Technology