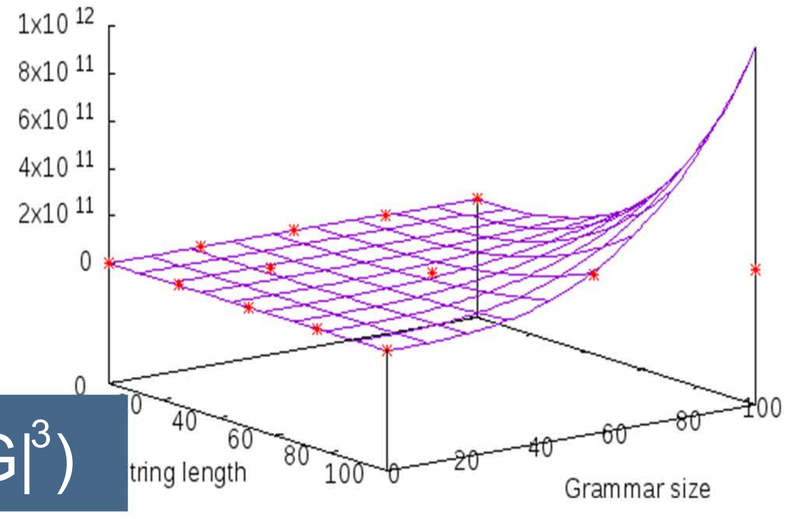
CYK Algorithm Adapted to the Penttonen Normal Form

Why? And how?

- as of now, no parser capable of processing context-sensitive grammars exists
- algorithm based on the CYK algorithm for context-free grammars in Chomsky normal form
- integrates context-sensitive rules in the form of AB → AC introduced by Penttonen normal form
- uses a versioning system to manage alternate parsing matrices created as the result of context conflict
- makes an unambiguous decision of whether the input string is a sentence of user-defined grammar

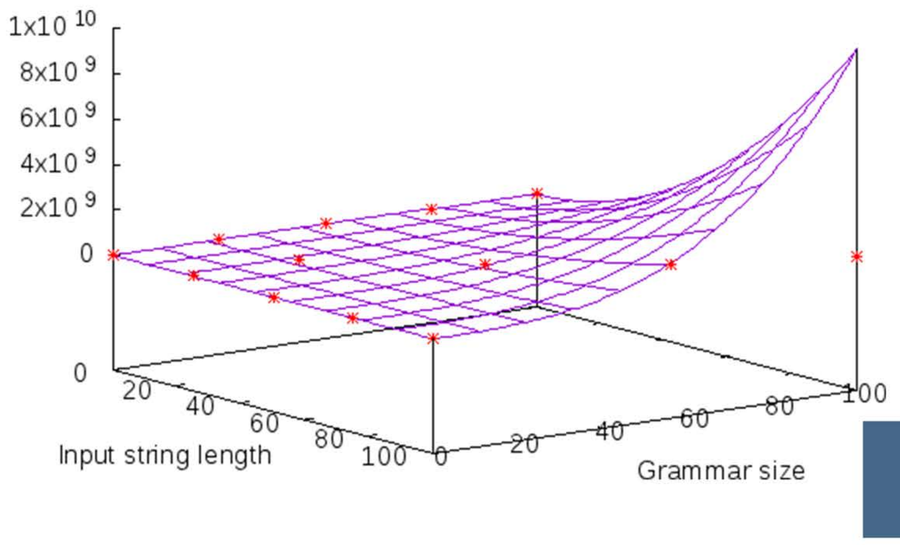
How many resources does it take?

- time complexity is similar to that of the original CYK algorithm, multiplied by the number of possible versions



$O(n^3|G|^3)$

- space complexity is affected mainly by the size of the grammar, as it affects both the number of possible versions as well as number of symbols in a set



$O(n^2|G|^3)$

How exactly?

- in case of a context-sensitive rule reduction, it adds the reduced nonterminal to the same set as the original symbol
- the set remembers the first right neighbour it is compared with to detect context collisions in time

CV[1,1] = {A}	CV[2,2] = {C}	CV[3,3] = {D}
he	*likes*	*rainbows*

- if a context collision is detected, the version is split into two
- the original version keeps the context-sensitive symbols, and the new version gets all subsequently reduces symbols

CV[1,2] = {} CV[2,3]

CV[1,1] = {A} CV[2,2] = {C, **B**} CV[3,3] = {D}

copy[1,2] = {} copy[2,3] = {E, F}

copy[1,1] = {A} copy[2,2] = {C} copy[3,3] = {D}

- in case of failure of currently used matrix, the version is abandoned, and a copy is chosen in its place

CV[1,3] = {S}

CV[1,2] = {} CV[2,3] = {E, F}

CV[1,1] = {A} CV[2,2] = {C} CV[3,3] = {D}

- if any such version succeeds, the string is **accepted**, otherwise it is **rejected.**