Turing Complete Systems

Members: Noah Sweet

When discussing Programming Languages it is hard to not describe at what point is something considered a Programming Language. I personally would love to look into the definition used to accomplish this problem. I use the word Turing Complete Systems vs just Turing Complete to account for systems like C++ Metaprogramming the move compiler--a compiler that only generates mov instructions (github). This ties in well with the concepts of grammar we have been discussing since a Turing Complete System will have a complete set of grammar rules that allow the user to accomplish a meaningful task.

I specifically want to discuss the odds and ends of weird Turing Complete Systems; researching both the bare minimum of a Turing Complete as well as some odd places it appears. Some languages I want to exemplify with this concept are C, C with the movcc, JavaScript, C++ Templates, and the C Preprocessor--which is not turing complete.

I believe this project will present a good ground for how to explain it in a presentation format and a deep look into how languages developed over the years with this as a consistent benchmark. Explaining why something itself is not turing complete, or why it is, can be a challenge as a decent amount of language knowledge is required to correctly understand its approach to solving this benchmark. The C++ template will probably be the biggest roadbump, but showing off the turing machine as well as application of the completeness should be doable in the time allocated.

I intend to work on this in the order I have listed above. C is a good place to start as it is the basis for many languages, syntax, and concepts we have in modern programming. I plan to write a turing machine in each language as best I can, or talk about useful aspect that come from the language being turing complete.

This concept is an important idea to keep in mind for the approach when writing a new language, since it is the benchmark we use to define when it is no longer an idea, but a language. It's quite odd to see a turing complete system come from generation code in another language. Researching and Discussing these concepts would greatly help me better understand the aspects and thought that went into designing a language.