Outline of the Talk

High-level overview: I will begin with a short monologue to familiarize the audience with stack languages and their semantics. I will keep this section short and non-formal, attempting to build up an intuition for how stack languages work. This portion will cover stack languages that support higher-order functions.

The rest of the talk will build up the formal semantics and type inference systems for a small stack-based language, starting with the minimal subset checked by Poial's first paper and ending with a system equivalent in power to Diggins'.

Part 1 - Background

1. Why bother with stack languages?
    a. Forth, Joy, Factor, etc.
2. Give an intuition for how stack languages work
3. Light overview of the history

Part 2 - Stack-Effect Calculi

1. Build up the system from Poial '90 to Horspool '93
    a. HPCL - HP Calculator language (numbers and numeric operators)
    b. CBL - HPCL + if statements
    c. MiniForth - CBL + stack shuffle words

Part 3 - Nested Tuple Systems

1. Continue building based on the stack-effect calculi
    a. HiForth - MiniForth - stack shuffle words + lambdas
    b. Show why we can't use basic stack effects
        i. Primary problem: no stack-polymorphic `call` operator
2. Develop Chris Okasaki's tuple embedding in Haskell
3. Explain Diggin's type syntax and show how it is equivalent to Okasaki
4. Continue building based on Diggin's system
    a. MiniJoy - HiForth + stack-polymorphic call