Practical Considerations in a Static Stack Checker

M. Anton Ertl, TU Wien

Static checking

- Stack depth checking
  Type checking

- Papers for more than 3 decades
  Prototypes
  no wide use

- Let's do something easy
  Check the stack depth

Static checking

- Stack depth checking
  Type checking

- Papers for more than 3 decades
  Prototypes
  no wide use

- Let's do something easy
  Check the stack depth
  harder than expected

What does a static Forth checker need?

language style	statically checked	no static checking
Example	StrongForth	Forth
Existing programs	written for checker	not written for checker
	pass	yet must pass
Programmers	accept compiler’s verdict	expect their idioms to pass

- (Almost) No false positives!

- False negatives ok

- Deal with unknown stack effects (`execute`)
  Assume that unknown stack effect is correct

Check against what?

- Stack effect comments?
  too much variation in practice
  possible future option

- Return stack effect `( R: -- )`

  ```
  : foo >r rot ;
  ```

- Control flow: Does the stack depth agree?

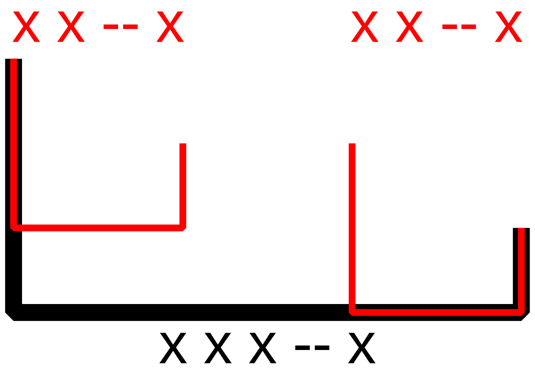
  ```
  do i loop
  if dup else drop then
  ```

- Unknown stack effect ⇒ possible false negative

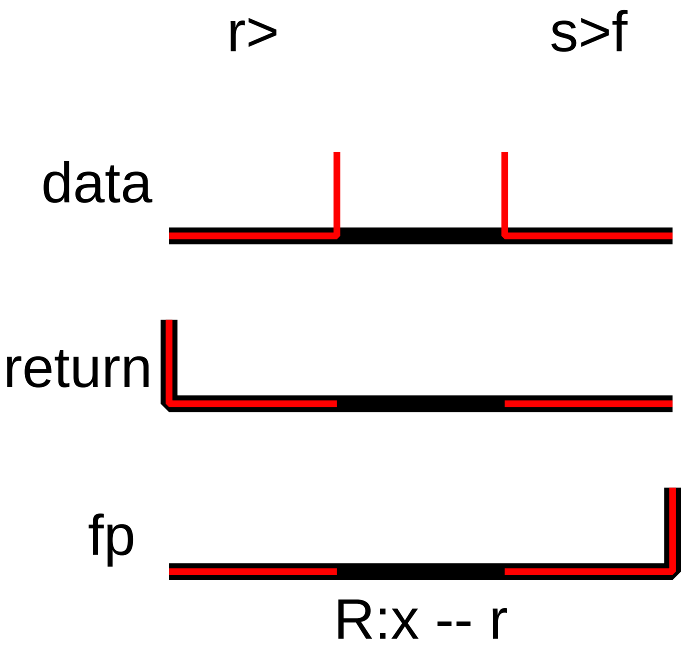
  ```
  if execute else drop then
  ```

How?

Sequential code



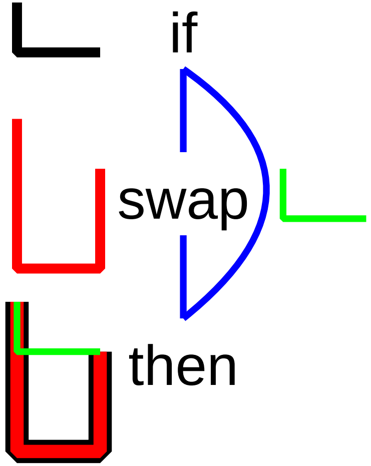
Multiple stacks



How?

Control flow

stack effects from start

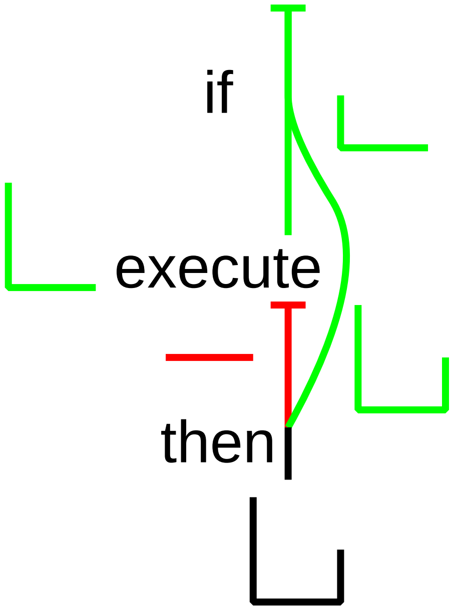


difference must be equal

flow direction does not matter

⇒ single pass

Unknown stack effects



unknown stack effect starts new anchor

propagate maximum depth

unify anchors on control flow meets

Implementation

- Leverage existing Gforth compiler:
  Add hooks in a few places

- Primitives:
  Hook in `peephole-compile,`
  Stack effect of nearly all primitives known
  also covers variables, constants, fields etc.

- colon definitions and `does>`-defined words:
  Hook in `compile,`, implementations for these
  for new definitions, store the stack effect from the checker
  for old definitions: *unknown* or set up with other mechanism

Implementation

- Control flow:
  Extra cell in control-flow stack item
  Hook `push-stack-state` (`begin if do` etc.)
  Hook `pop-stack-state` (`until then loop` etc.)
  New anchor after unconditional branch (`again ahead exit`)

- Unknown stack effect:
  propagate maximum depth
  Start new anchor

Conclusion

- Static checking: lots of research, little use
- Work with (partly) legacy code ⇒ no false positives!
  Assume that any unknown stack effect is correct
  Don't rely on stack effect comments
  Check return stack balance and on control flow meet
- Checker deals with sequences, multiple stacks, control flow

- Unknown stack effects introduce new anchors
  Unified with existing anchors on control flow meets
- Implement by hooking into existing Forth compiler
- Status: anchors not yet functional