Exploiting Automatic Flow Analysis

Jens Bendisposto
Stefan Hallerstede
Michael Leuschel

Outline

- Automatic Flow Analysis
  - for Code Generation
  - for Model Checking (Liveness)
- Example: Extended GCD

Flow Analysis

Flow-Analysis

- Arises from the POs
  - $I \wedge G \wedge S \wedge P \Rightarrow H$
  - $I \wedge G \wedge S \wedge Q \Rightarrow \neg H$
- G and H are the Guards of events g and h
- P and Q are enabling and disabling conditions
- We need to calculate P and Q

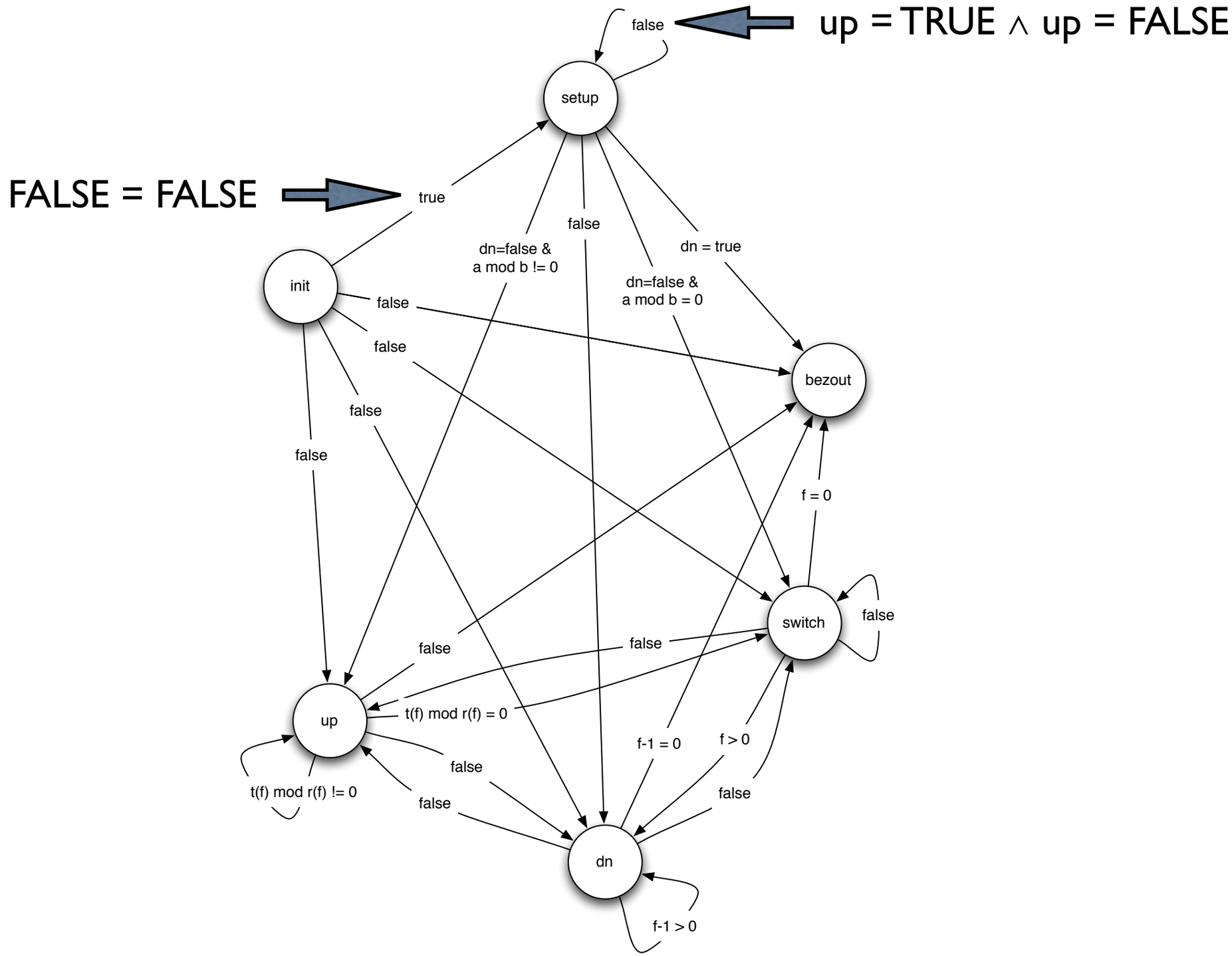
Flow-Analysis

- Let Q = ¬P and P = [S]H
- [S]H is the weakest precondition that guarantees to enable h after g occured
- Both POs obviously hold for P and Q

Step 1: Enable Graph

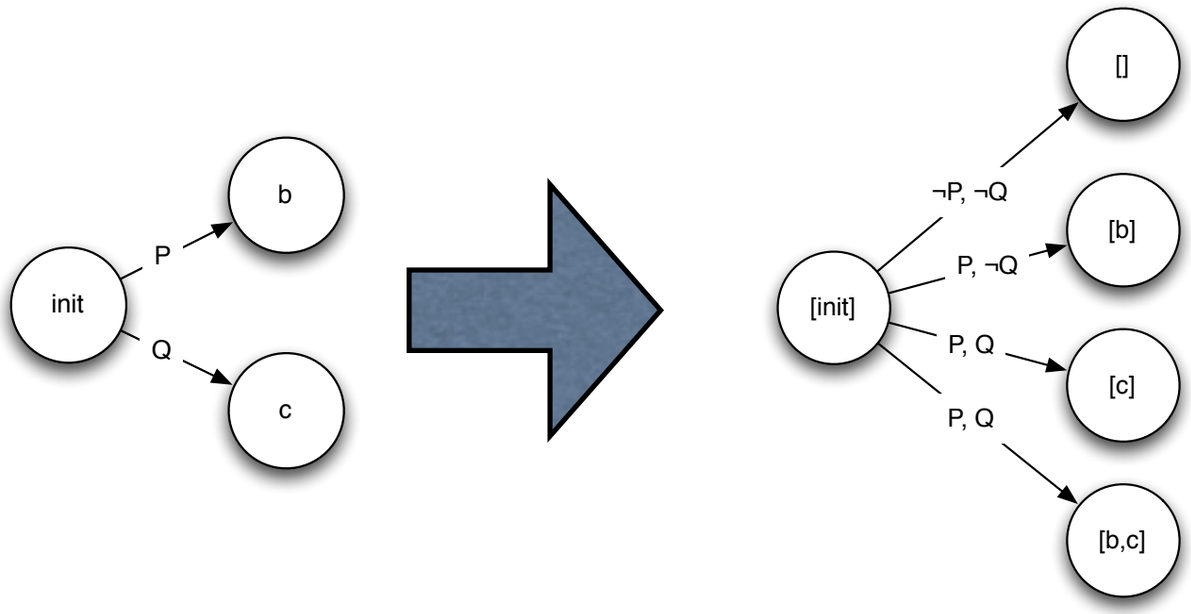
- Create a directed Graph containing all events as nodes
- For each pair of events g,h we calculate read(h) and write(g)
- If read(h) ∩ write(g) ≠ ∅, there is an edge from g to h
- This edge is labeled with [S]H
- Simplify [S]H (use axioms, theorems, invariants)

Example: GCD


up = TRUE ∧ up = FALSE
FALSE = FALSE
false
setup
true
init
false
dn=false &
a mod b != 0
false
dn=false &
a mod b = 0
dn = true
bezout
false
false
false
false
f = 0
switch
false
false
false
up
t(f) mod r(f) = 0
f-1 = 0
f > 0
false
t(f) mod r(f) != 0
false
false
dn
f-1 > 0

Step 2: Flow

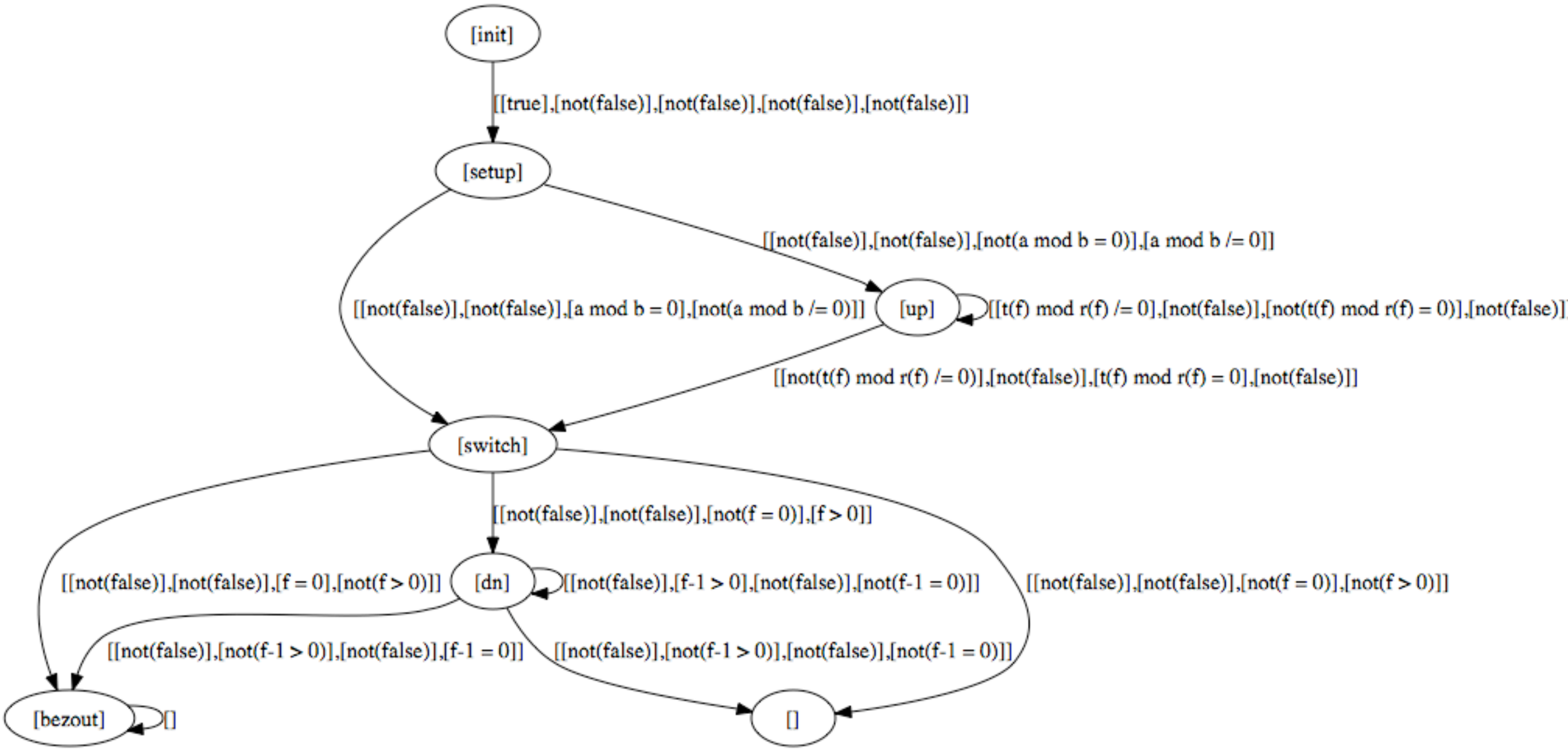
- Expand the Graph starting from a state [init]
- [x,y] means events x and y are enabled
- For each enabled event: Combine all predicates from the Enable Graph



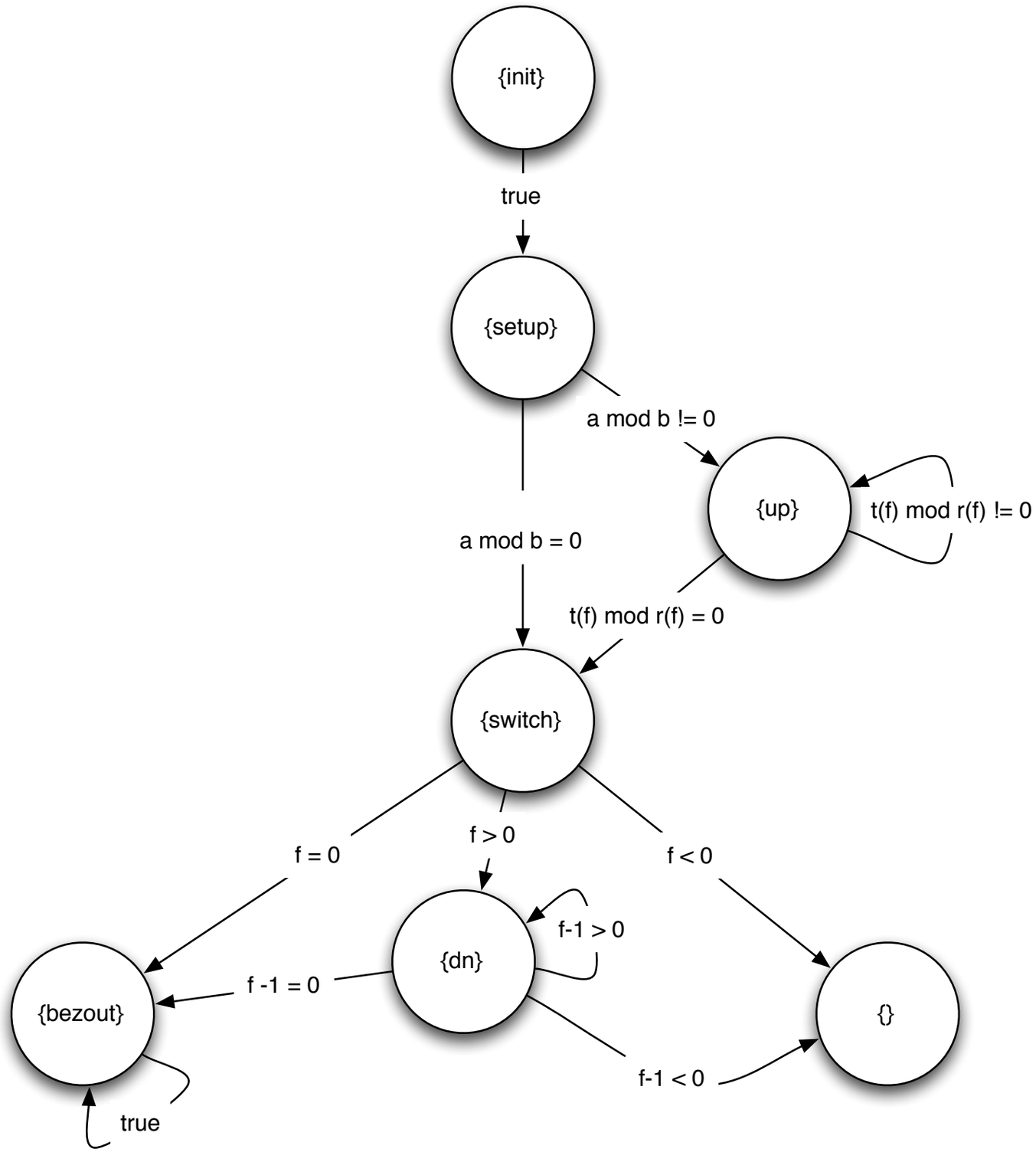
Step 2: Flow

- Can blow up exponentially
- Heavily depends on good simplification
- Evidence that it works (at least) for deterministic systems

Result from a prototype



Pretty Printed



A simple Prover

```
contradicts(false,_).
contradicts(not(true),_).
contradicts(X,not(X)).
contradicts('dn=false','dn=true').
contradicts(not('dn=false'),not('dn=true')).
contradicts('a mod b /= 0','a mod b = 0').
contradicts(not('a mod b /= 0'),not('a mod b = 0')).
contradicts('t(f) mod r(f) = 0','t(f) mod r(f) /= 0').
contradicts(not('t(f) mod r(f) = 0'), not('t(f) mod r(f) /= 0')).
contradicts('f = 0','f > 0').
contradicts('f-1 = 0','f-1 > 0').
```

Model Checking

Speed up MC

- To calculate sucessor states ProB evaluates the Guards
- From Flow Analysis
  - We know, that some events are always disabled
  - No need to check these guards
- For all other events, it can be better to evaluate P instead of the guard
- Enable Graph is sufficient!

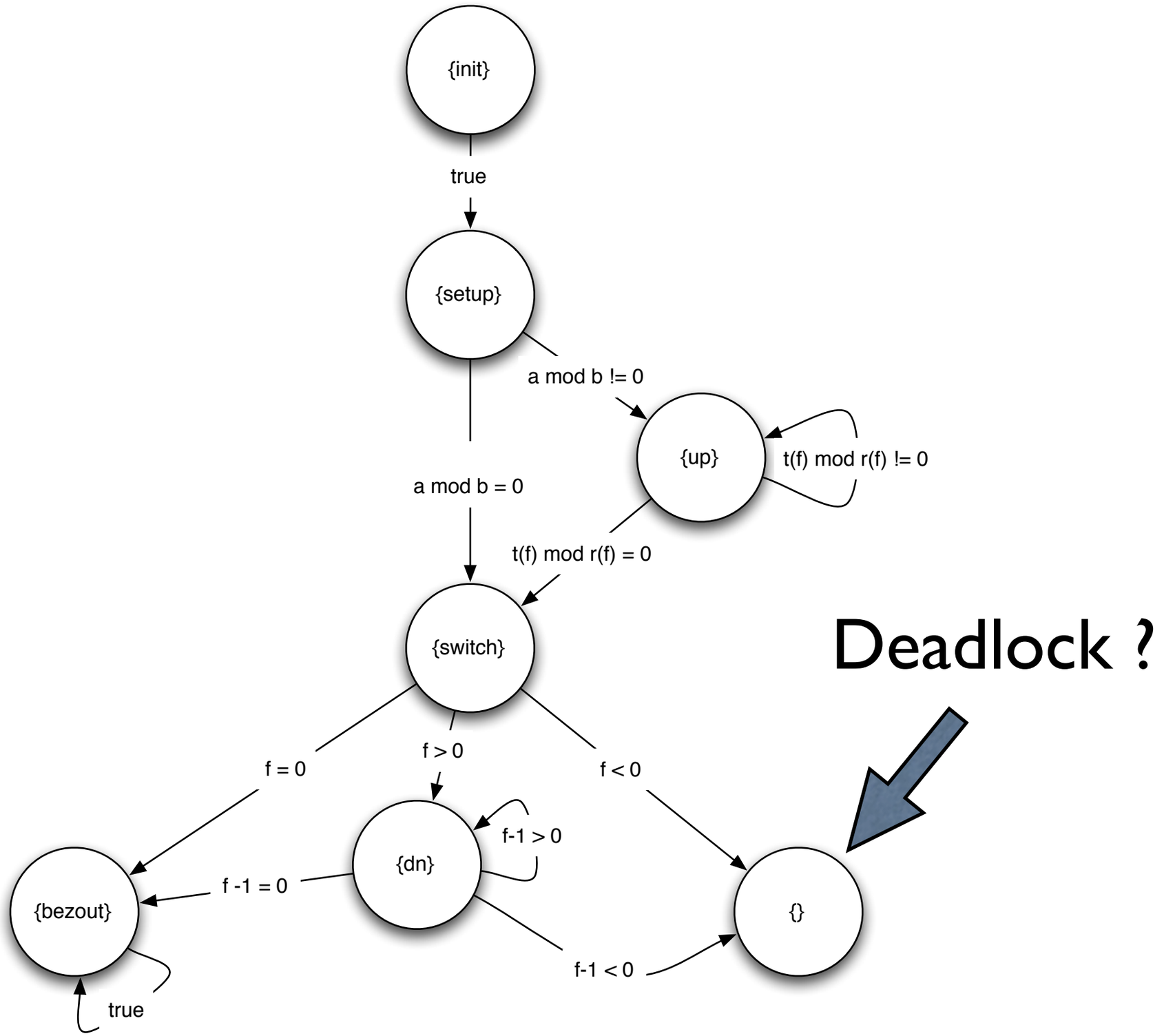
Directed Model Checking

- From Proof, we know which event E might break an Invariant
- Use Flow Information to find a path where E is enabled
- Needs Flow Graph

Liveness Analysis

- Deadlock Detection
- If the state [] is absent, the system is deadlock free
- If [] is present, feasibility of the paths should be checked
- Spurious alerts are possible

Liveness Analysis



Deadlock ?

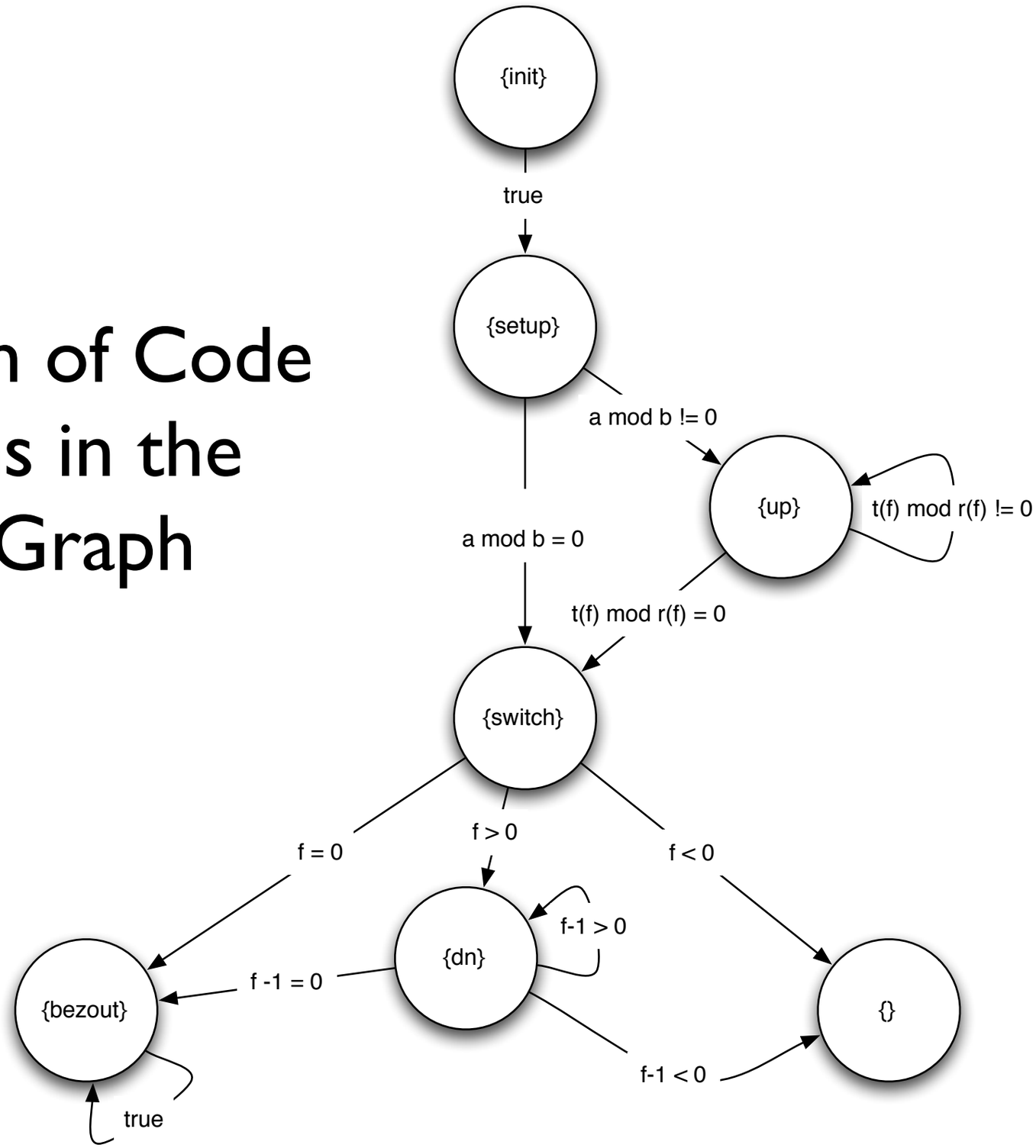
Code Generation

Targets

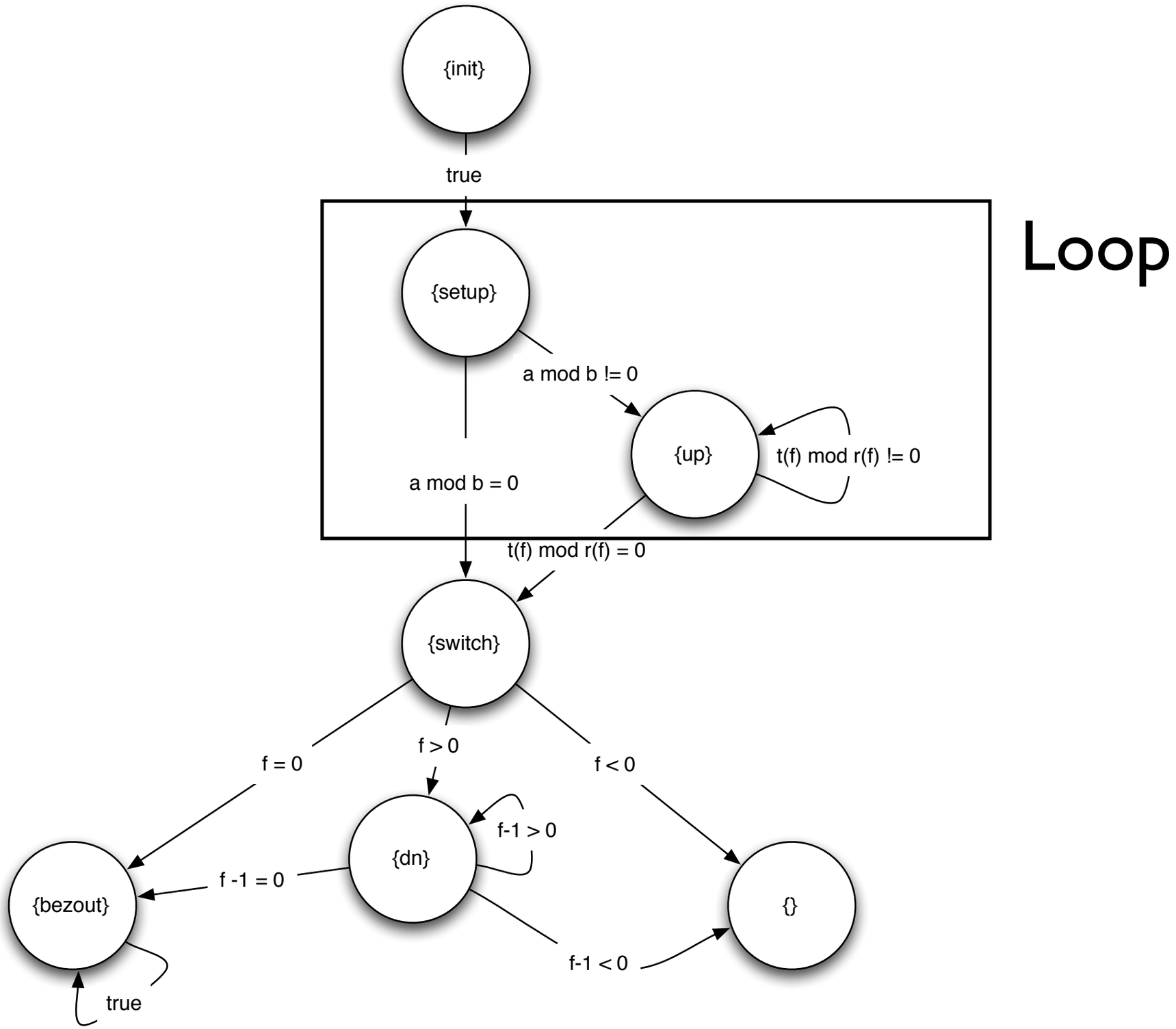
- Develop Code for AUTOSAR
- Realtime
- Concurrent blocks of sequential code

Code Generation

Detection of Code Patterns in the Flow Graph

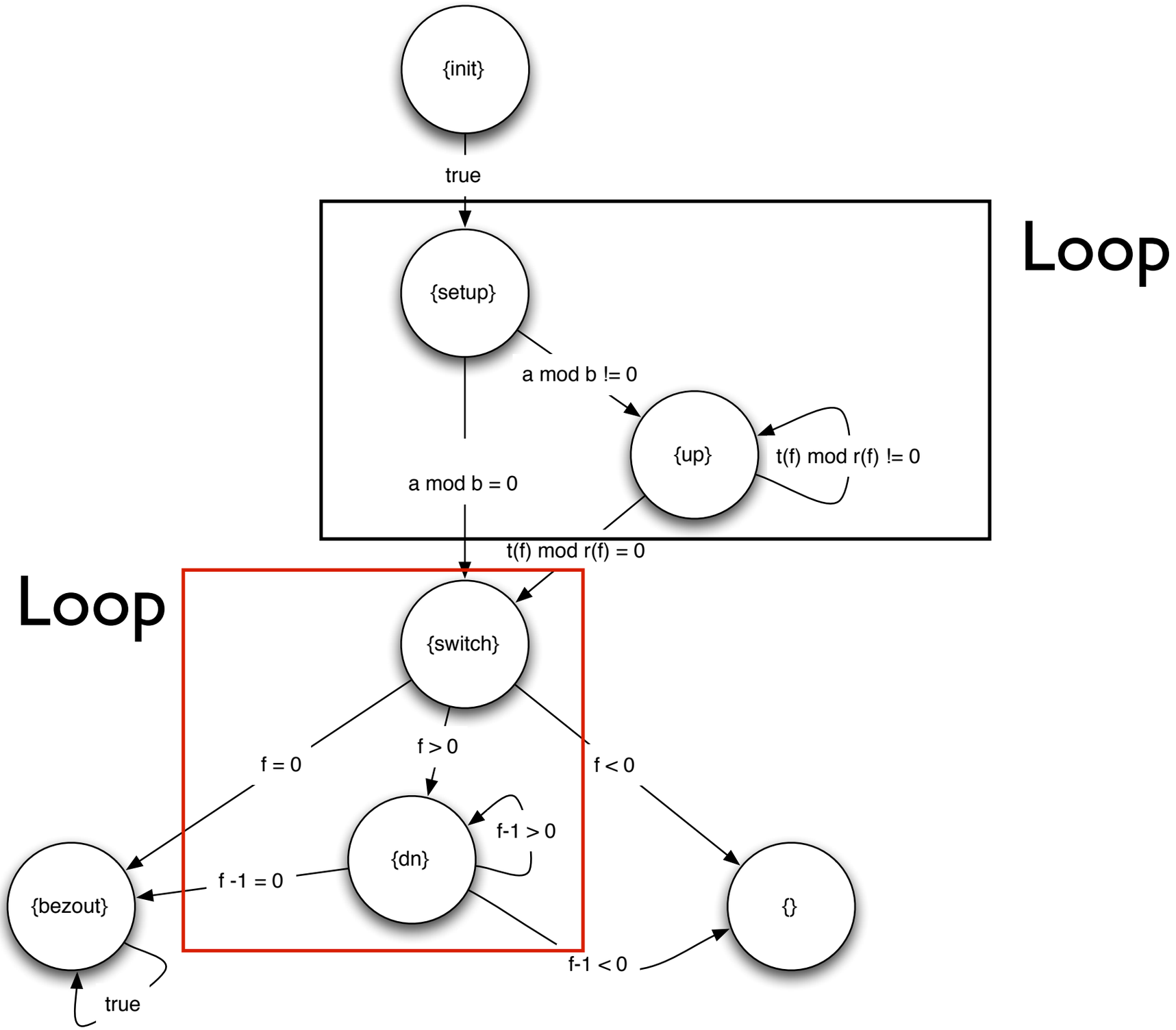


Code Generation



Loop

Code Generation



Loop

Loop

Code Generation

- Needs the Flow Graph
- Simplifier removes control variables

Benchmarks

✓ Extended GCD

✓ List Reversal

? Quicksort

? Schorr-Wait