Making Cyclic Circuits Acyclic

Stephen A. Edwards

Department of Computer Science,
Columbia University

www.cs.columbia.edu/˜sedwards

sedwards@cs.columbia.edu

Goal

Given a *constructive* cyclic circuit, create an equivalent acyclic circuit.

Applications:

- Replaces the resynthesis portion of Esterel's **`sccausal`**.
- Can be adapted for Esterel software synthesis.
- Useful when solving large systems of equations.

Related Work

Malik's algorithm, 1993

- Remove enough gates to make the graph acyclic
- Make that many copies of the circuit

Bourdoncle, 1993

- Recursive SCC decomposition
- Remove a single gate at each step

Edwards' Thesis, 1997

- Bourdoncle variant
- SCC decomposition, may remove two or more gates at each step

Proposed Algorithm

1. Determine all possible schedules

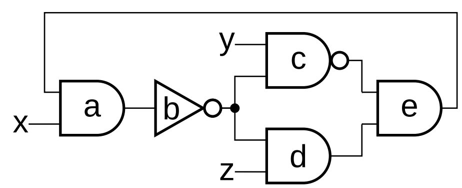
   (Each a circuit fragment)

2. Merge (overlay) fragments to generate a small circuit

Advantage: takes into account actual circuit behavior, not approximation thereto.

Disadvantages: may be too many schedules and optimal merging appears difficult

Example Circuit



Controlling Value

A *controlling value* is a 0 input on an AND gate, a 1 on an OR.

In constructive logic, this value causes the gate to ignore the rest of its inputs.

$\wedge$	$\perp$	0	1
$\perp$	$\perp$	0	$\perp$
0	0	0	0
1	$\perp$	0	1

$\vee$	$\perp$	0	1
$\perp$	$\perp$	$\perp$	1
0	$\perp$	0	1
1	1	1	1

Theorem

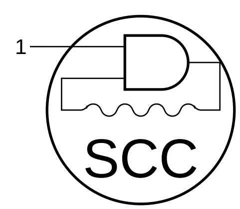
For an SCC to be constructive, at least one of its external inputs must be take a controlling value.

Proof by contradiction: if all inputs are non-controlling, by definition, the output of each gate is only affected by values within the SCC. These are initially all $\perp$, meaning all outputs are all $\perp$ and therefore the non-constructive least fixed point. ∎

Consequence: Any possible constructive schedule must start at a controlling value at an input.

Consequence: Recursive SCC decomposition obtained by injecting all possible controlling values will find all possible constructive schedules.

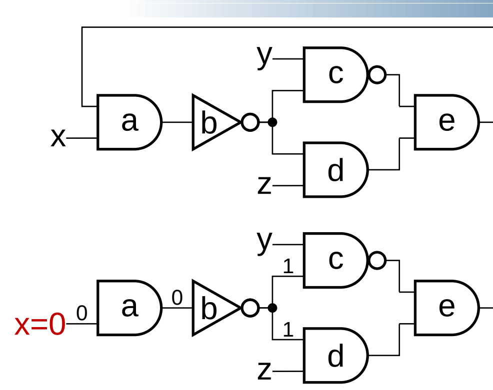
Intuition



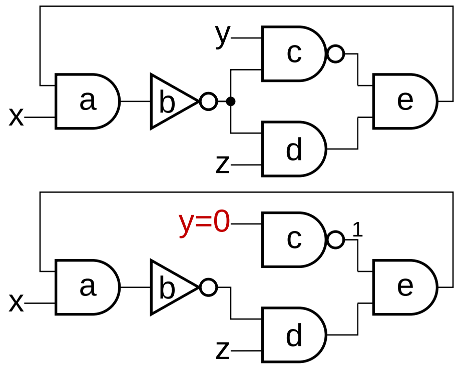
If every such external input was set to 1 (all AND gates), the SCC would have a fixed point of all $\perp$.

Thus, at least one of these external inputs to 0. This condition is necessary, but not sufficient.

Finding all schedules (step 1)

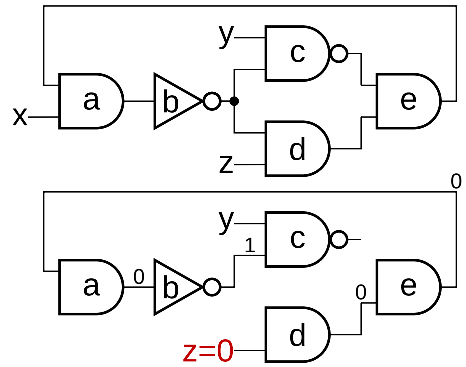


Finding all schedules (step 2)



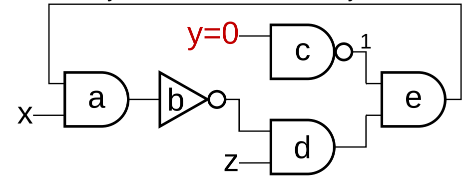
Still cyclic: Deal with it later

Finding all schedules (step 3)



Finding all schedules (step 4)

We found two acyclic schedules and one cyclic schedule:

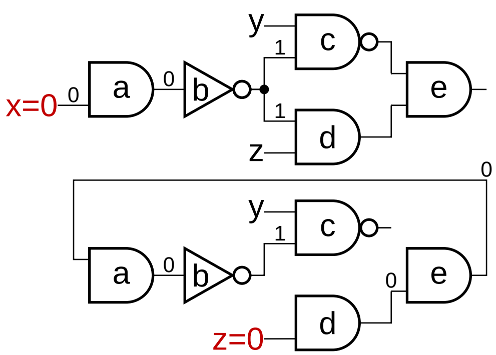


The three inputs to this are x, z, and the output of gate c.

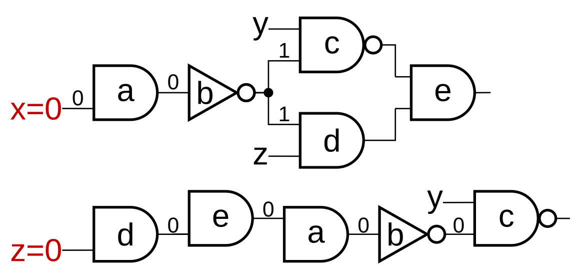
However, x=0 and z=0 were earlier found acyclic. And the output of gate c is fixed at 1 since y=0.

We are done: we won't get any *other* acyclic schedules from this.

Merging Schedules (part 1)



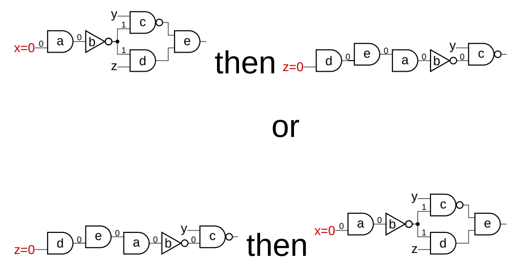
Merging Schedules (part 2)



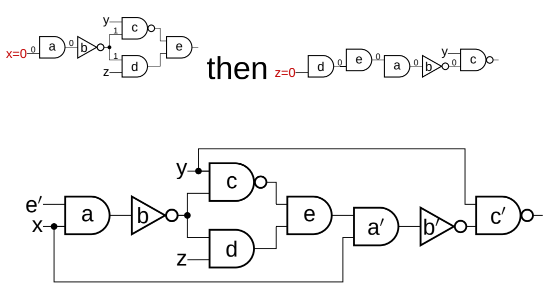
Second is same as before, just unrolled.

Merging Schedules (part 3)

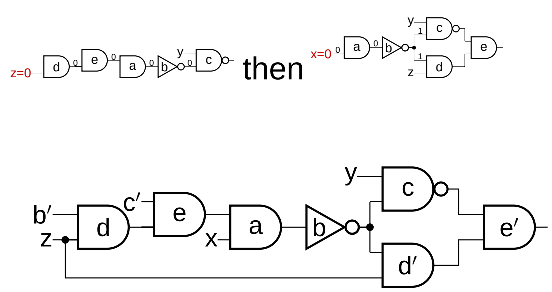
Two choices:



Merging Schedules (part 4)

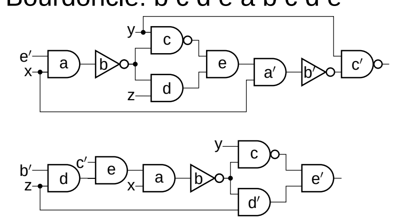


Merging Schedules (part 5)



Schedule Comparison

Dumb: abcdeabcdeabcdeabcdeabcde	= 25
Bourdoncle: b c d e a b c d e	= 9

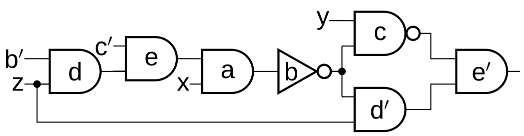


= 8

= 7

Simplifying the circuit

The second one,



is definitely smaller (seven gates versus eight).

What values should the b′ and c′ inputs take?

Knaster/Tarski/Kleene/Cousot theorem says they should be $\perp$. But it's difficult to build circuits that manipulate $\perp$.

Can we do better?

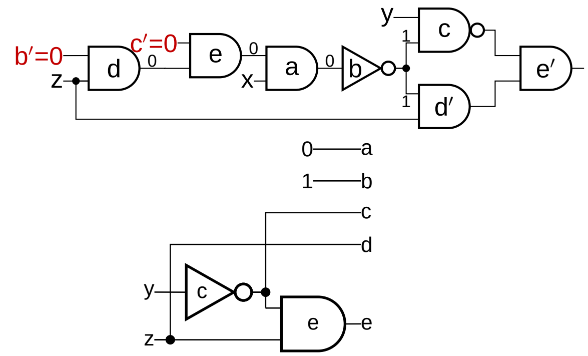
Theorem

Formerly internal signals that have become inputs may be set to either 0 or 1 without changing the function.

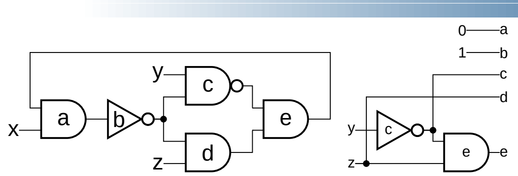
Proof. The least-fixed-point function F (i.e., the acyclic circuit) is monotonic, and is guaranteed to be causal, i.e., the least fixed point never contains $\perp$ values. Since F is monotonic and $\perp \sqsubseteq X$ by definition, $F(\perp) \sqsubseteq F(X)$. However, $F(\perp)$ is the least fixed point and fully defined, therefore we must have $F(\perp) = F(X)$. ■

Consequence: We can greatly simplify the circuit.

Simplifying the Circuit



Did we get it right?



x	y	z	a	b	c	d	e
0	-	-	0	1	¬y	z	¬y ∧ z
-	-	0	0	1	¬y	0	0
1	0	1	⊥	⊥	1	⊥	⊥
1	1	1	⊥	⊥	⊥	⊥	⊥

Conclusions

A procedure for building an acyclic circuit from a cyclic one

Can produce very compact circuits, especially after simplification

Smaller than Malik or Bourdoncle

Basic idea: enumerate schedules, merge them

Potential problems: too many schedules, non-optimal merging

What I haven't shown you: (complex) details of the search algorithm.