Examples:
Modelling formalisms, temporal logics, model checking

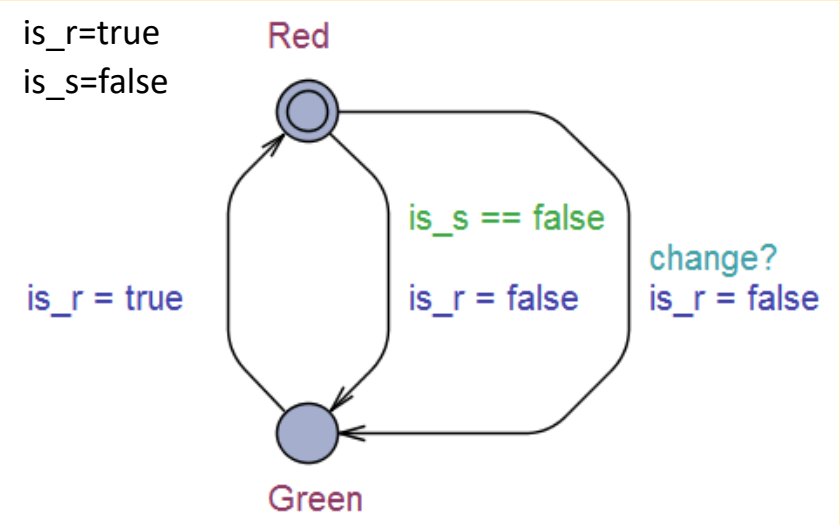
dr. István Majzik
BME Department of Measurement and Information Systems

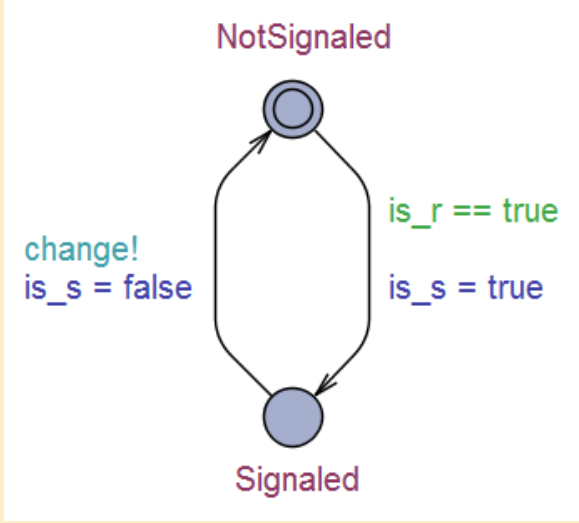
Interpretation of formal models

Interpretation of automata models

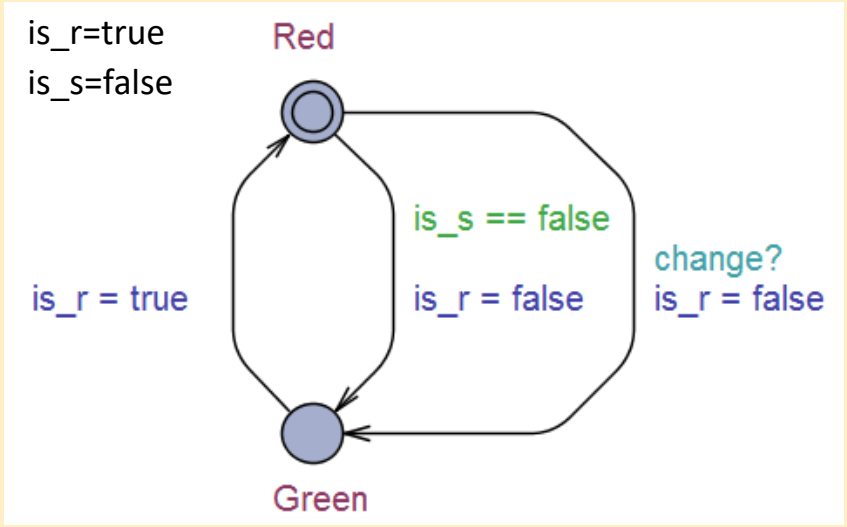
Consider the following two automata models (constructed in the UPPAAL tool) that model the behavior of a traffic light and a pedestrian. In the initial state is_r=true and is_s=false.

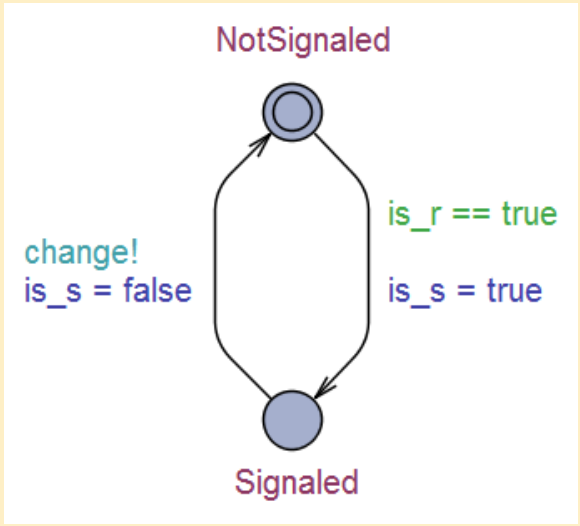
- Draw the Kripke structure corresponding to the whole system, i.e., the reachable combinations of the states of the traffic light and the pedestrian, and the related transitions!
  Label each combined state with the names of the states that it represents (you can use the initial letters of the states).

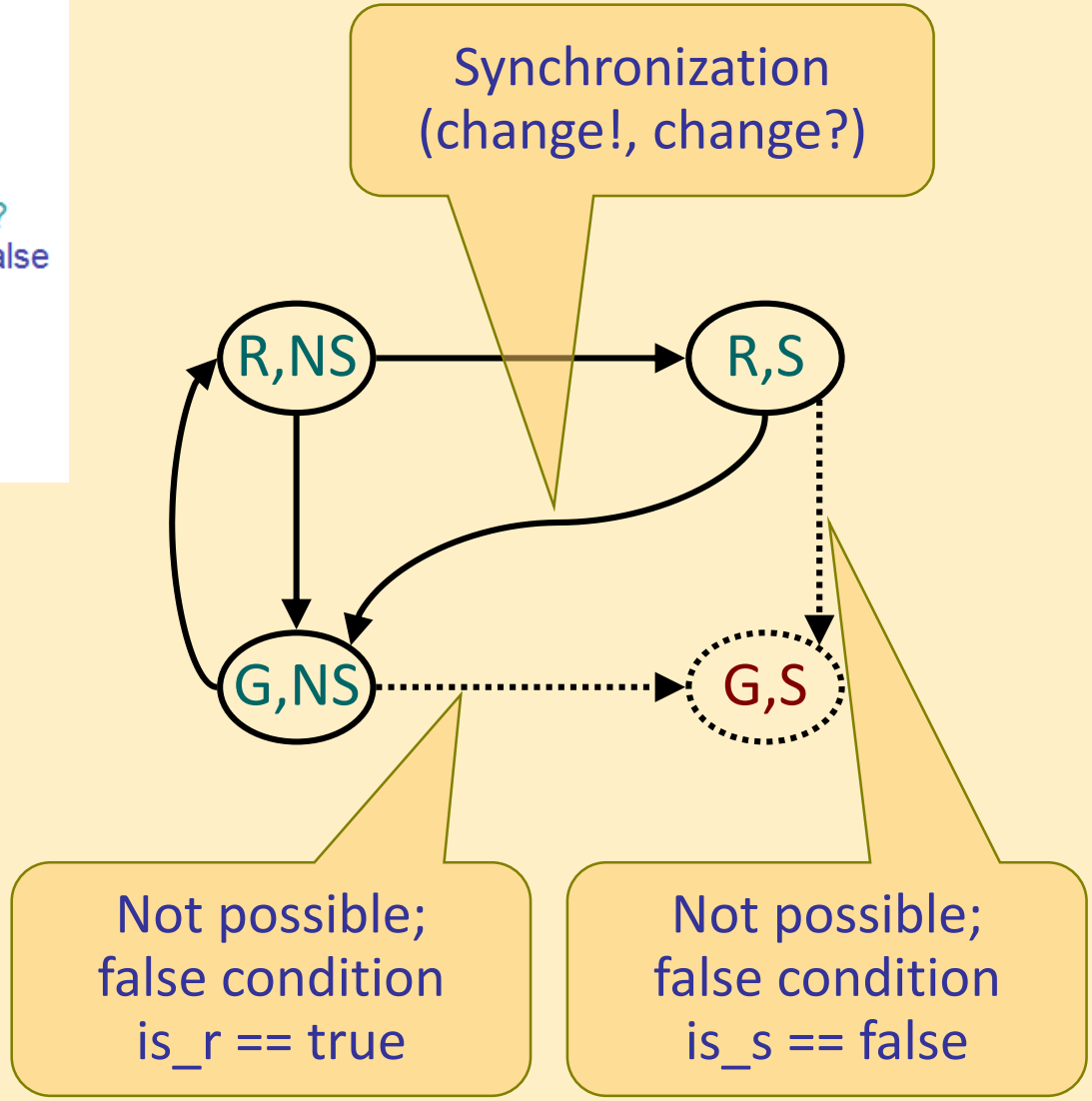




Interpretation of automata models – Solution







Formalization of properties using temporal logics

Theoretical questions

Argue if the following LTL equivalences are correct or not:

1. F (Start $\vee$ Stop) $\equiv$ (F Start) $\vee$ (F Stop)
2. G Stop $\equiv$ not F (not Stop)

Argue if the following CTL equivalences are correct or not:

1. AF (Start $\vee$ Stop) $\equiv$ (AF Start) $\vee$ (AF Stop)
2. AF (Start $\wedge$ Stop) $\equiv$ (AF Start) $\wedge$ (AF Stop)
3. EF (Start $\wedge$ Stop) $\equiv$ (EF Start) $\wedge$ (EF Stop)

Argue if the following formula are syntactically correct in CTL or not:

1. A (X Stop $\vee$ F Start)
2. A (Stop U (AX Start))

Theoretical questions – Solution 1/3

Argue if the following LTL equivalences are correct or not:

1. F (Start $\vee$ Stop) ≡ (F Start) $\vee$ (F Stop)
   Correct: Left side of the equivalence follows from the right side of the equivalence and vice versa.

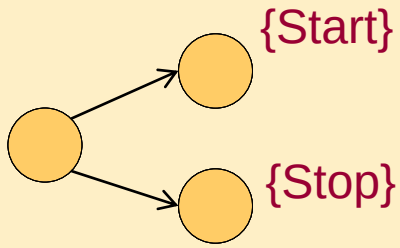
2. G Stop ≡ not F (not Stop)
   Correct: Left side of the equivalence follows from the right side of the equivalence and vice versa.

Theoretical questions – Solution 2/3

Argue if the following CTL equivalences are correct or not:

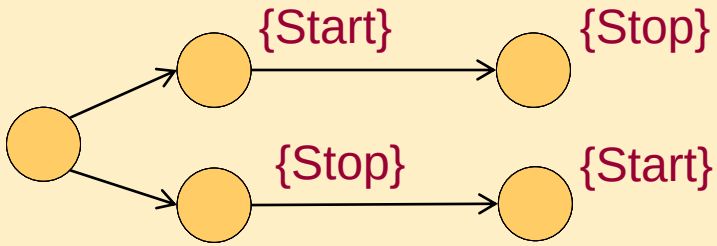
1. AF (Start $\vee$ Stop) $\equiv$ (AF Start) $\vee$ (AF Stop)

   Counterexample 1: Left side of the equivalence is true but right is not.



2. AF (Start $\wedge$ Stop) $\equiv$ (AF Start) $\wedge$ (AF Stop)

   Counterexample 2: Right side of the equivalence is true but left is not.



3. EF (Start $\wedge$ Stop) $\equiv$ (EF Start) $\wedge$ (EF Stop)

   Counterexample 1 above: Right side is true but left is not.

Theoretical questions – Solution 3/3

Argue if the following formula are syntactically correct in CTL or not:

1. A (X Stop $\vee$ F Start)
   Not syntactically correct in CTL: There is Boolean operator $\vee$ between two path expressions X Stop and F Start
   (and this is not allowed in CTL)

2. A (Stop U (AX Start))
   Correct: Here the operator AU is applied on two state expressions Stop and AX Start

Requirement formalization: Railway crossing

- We model the behavior of a railway crossing signal with the following atomic labels:

  {off, white, red}

- The behavior of the car driver arriving at the crossing is modeled with the following atomic labels:

  {arriving, looking, stopping, crossing}

- Use LTL expressions to formalize the following properties which apply to the behavior of the driver in every case (continuously):

  1. If the signal is off, the driver will be looking and then in the next moment either stopping or crossing.
  2. The driver will eventually cross the crossing.
  3. If upon arrival, the signal is red, the driver will not cross until the signal is white.

Railway crossing – Solution

- Labels for the signal:
  {off, white, red}
- Labels for the driver:
  {arriving, looking, stopping, crossing}
- LTL expressions formalizing the properties:

  Note: "apply … in every case (continuously)": initial **G** operator

  1. If the signal is off, the driver will be looking and then in the next moment either stopping or crossing.
     $\mathbf{G}\ (\text{off} \rightarrow (\text{looking} \wedge \mathbf{X}\ (\text{stopping} \vee \text{crossing})))$
  2. The driver will eventually cross the crossing.
     $\mathbf{G}\ \mathbf{F}\ \text{crossing}$
  3. If upon arrival, the signal is red, the driver will not cross until the signal is white.
     $\mathbf{G}\ ((\text{arrival} \wedge \text{red}) \rightarrow ((\neg\ \text{crossing})\ \mathbf{U}\ \text{white}))$

Requirement formalization: Server room

- We model the states of a server performing complex simulation with the following atomic labels:
  {off, waiting, warm-up, simulation}
- The air-conditioning system is modeled with the following atomic labels:
  {stand-by, normal, maximal}
- Use LTL expressions to formalize the following properties which apply to the behavior of the server continuously:
  1. If in any moment the simulation is performed with the air-conditioning system being in the stand-by state, then in the next moment, the server will move to the waiting state.
  2. Eventually, the simulation will be started.
  3. The simulation can be performed only if it was preceded by a warm-up phase with the air-conditioning in the normal state.

Server room – Solution

- Labels for the server:
  {off, waiting, warm-up, simulation}
- Labels for the air-conditioning system:
  {stand-by, normal, maximal}
- LTL expressions formalizing the properties:

  Note: "apply … continuously": initial **G** operator

  1. If in any moment the simulation is performed with the air-conditioning system being in the stand-by state, then in the next moment, the server will move to the waiting state.
     $\mathbf{G}\ ((\text{simulation} \wedge \text{stand-by}) \rightarrow \mathbf{X}\ \text{waiting})$
  2. Eventually, the simulation will be started.
     $\mathbf{G}\ \mathbf{F}\ \text{simulation}$
  3. The simulation can be performed only if it was preceded by a warm-up phase with the air-conditioning in the normal state.
     $\mathbf{G}\ ((\mathbf{X}\ \text{simulation}) \rightarrow (\text{warm-up} \wedge \text{normal}))$

Model checking algorithms

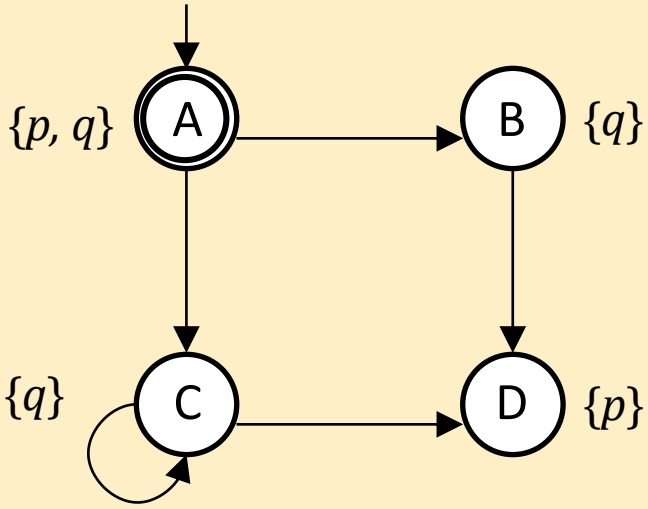
Checking CTL using iterative labeling

Consider the Kripke structure given below.

- Check if the following CTL expression holds from the initial state using the iterative labeling algorithm presented in the lectures:

$$\mathbf{A}(p\ \mathbf{U}\ (\mathbf{EX}\ \neg q))$$

For each iteration give the expression that is currently used for labeling and enumerate the states that are labeled!

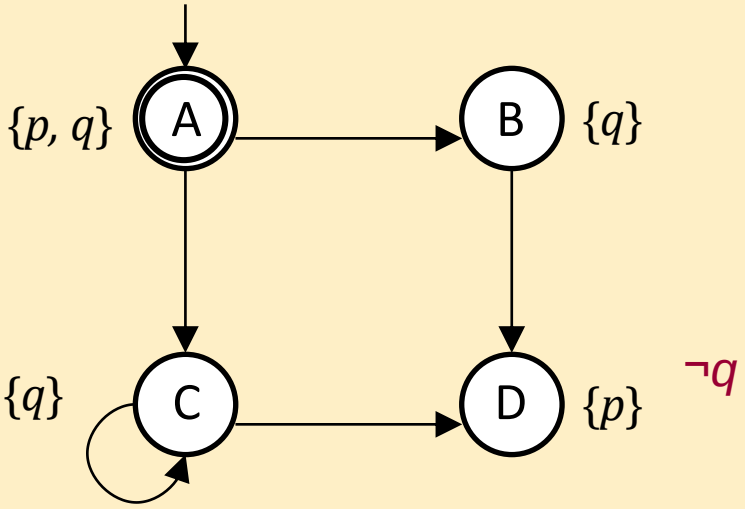


Checking CTL using iterative labeling – Solution 1/4

Check if the following CTL expression holds from the initial state using the iterative labeling algorithm:

A(*p* **U** (**EX** ¬*q*))

1. Labeling D with ¬*q*

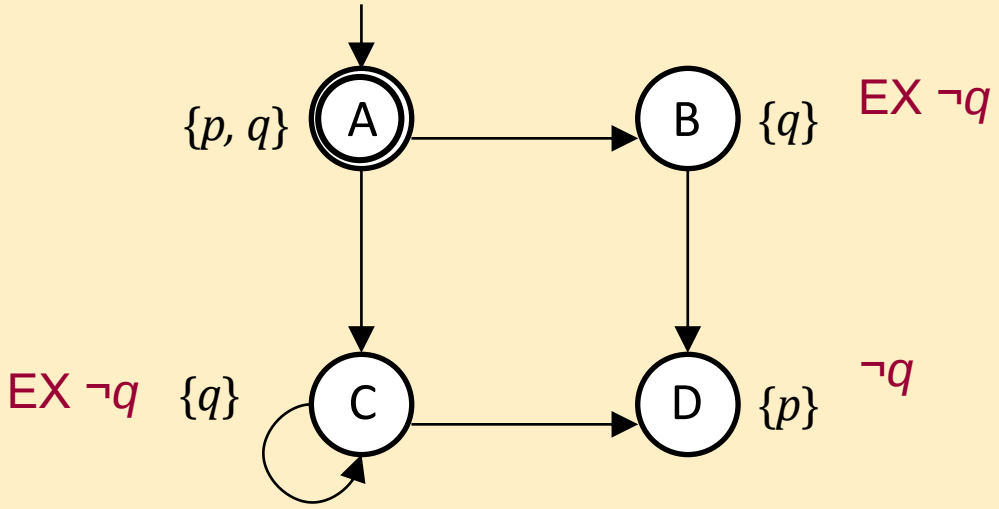


Checking CTL using iterative labeling – Solution 2/4

Check if the following CTL expression holds from the initial state using the iterative labeling algorithm:

A(*p* **U** (**EX** ¬*q*))

1. Labeling D with ¬*q*
2. Labeling B and C with **EX** ¬*q*

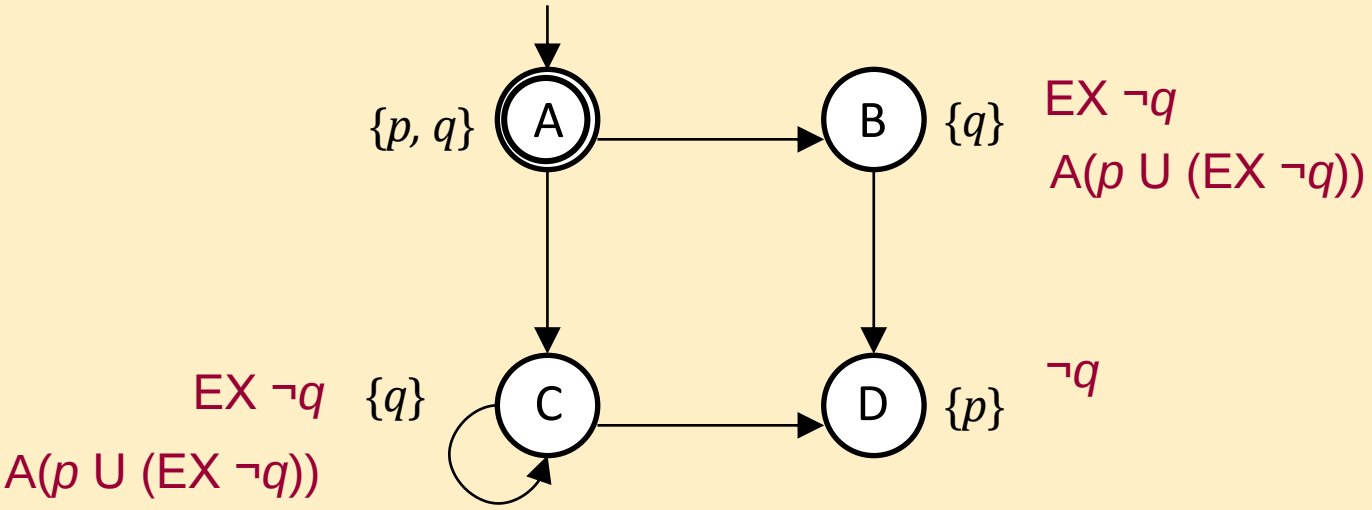


Checking CTL using iterative labeling – Solution 3/4

Check if the following CTL expression holds from the initial state using the iterative labeling algorithm:

A(*p* **U** (**EX** ¬*q*))

1. Labeling D with ¬*q*
2. Labeling B and C with **EX** ¬*q*
3. Labeling B and C with **A**(*p* **U** (**EX** ¬*q*))

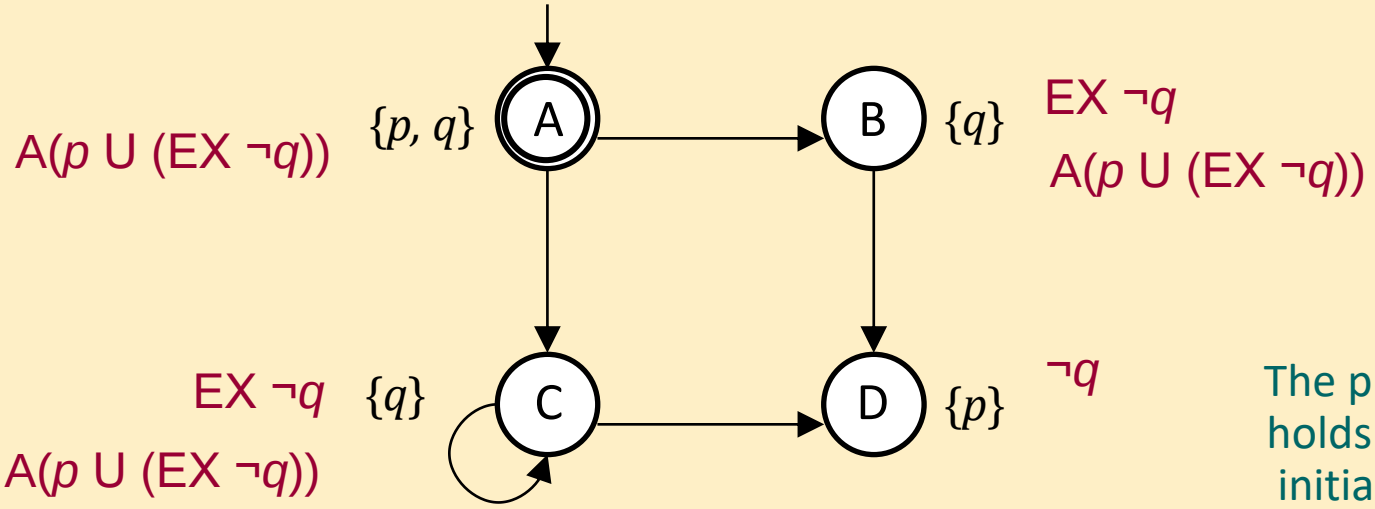


Checking CTL using iterative labeling – Solution 4/4

Check if the following CTL expression holds from the initial state using the iterative labeling algorithm:

A(*p* **U** (**EX** ¬*q*))

1. Labeling D with ¬*q*
2. Labeling B and C with **EX** ¬*q*
3. Labeling B and C with **A**(*p* **U** (**EX** ¬*q*))
4. Labeling A with **A**(*p* **U** (**EX** ¬*q*)) End of the iteration.



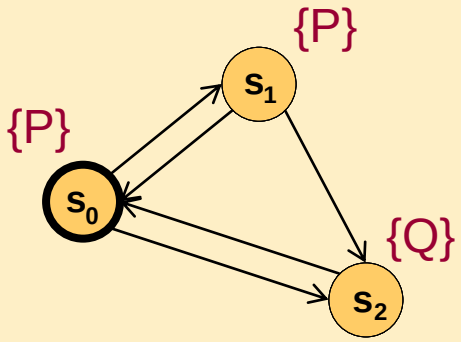
The property holds for the initial state.

Model checking with the tableau method

Consider the Kripke structure on the right.

Perform the model checking of the following formula with the tableau method:

$$\neg (P \; U \; Q)$$

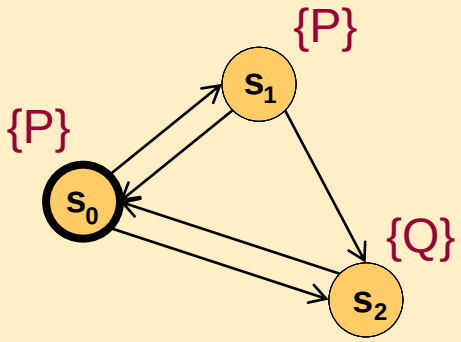


Model checking with the tableau method

Consider the Kripke structure on the right.

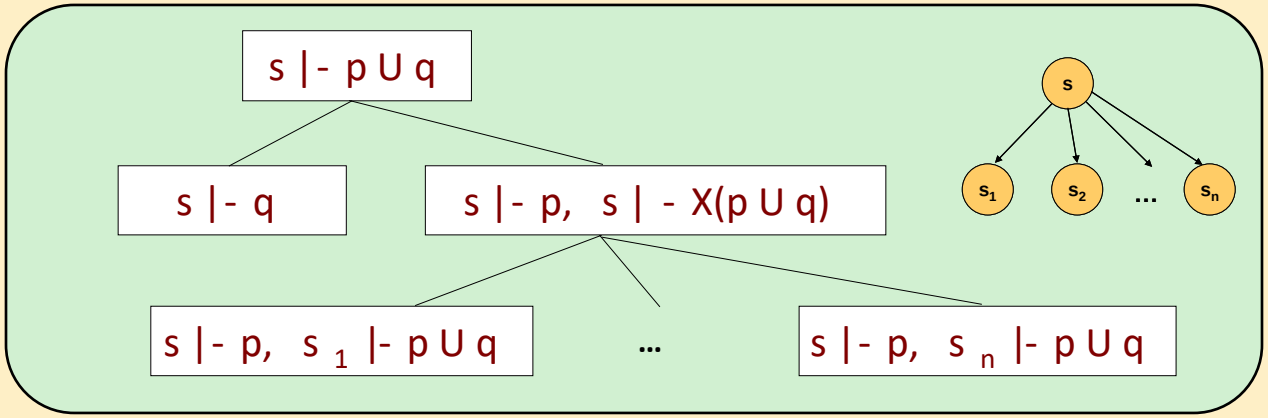
Perform the model checking of the following formula with the tableau method:

$\neg$ (P U Q)



Things to know:

- Negation (to look for counterexamples): (P U Q)
- Tableau rule: (P U Q) = Q $\vee$ (P $\wedge$ X(P U Q))

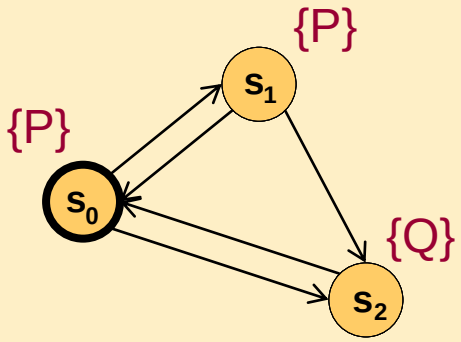


Model checking with the tableau method

Consider the Kripke structure on the right.

Perform the model checking of the following formula with the tableau method:

$$\neg (P \; U \; Q)$$

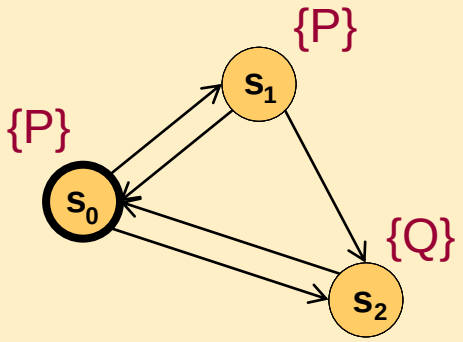


Things to know:

- Negation (to look for counterexamples): $(P \; U \; Q)$
- Tableau rule: $(P \; U \; Q) = Q \vee (P \wedge X(P \; U \; Q))$
- Contradicting branch if:
  - Atomic proposition does not hold in a state
  - X operator but no outgoing transition from a state
  - Loop in the tableau with P, but without Q
- Satisfying branch (here: giving counterexamples) if:
  - Only atomic propositions, and all of them hold in the state
  - Loop in the table without contradiction

Model checking with the tableau method

Tableau construction:



s_0 |- P U Q

s_0 |- Q

s_0 |- P, s_0 |- X(P U Q)

Contradicting branch

s_1 |- P U Q

s_2 |- P U Q

s_1 |- Q

s_1 |- P, s_1 |- X(P U Q)

s_2 |- Q

s_2 |- P, s_2 |- X(P U Q)

Contradicting branch

s_0 |- P U Q

s_2 |- P U Q

Satisfying branch

Contradicting branch

Contradicting branch: Existing tableau node, loop back with P only

Existing tableau node (but not a loop!)

Counterexamples for ¬(P U Q): s_0, s_2 and s_0, s_1, s_2

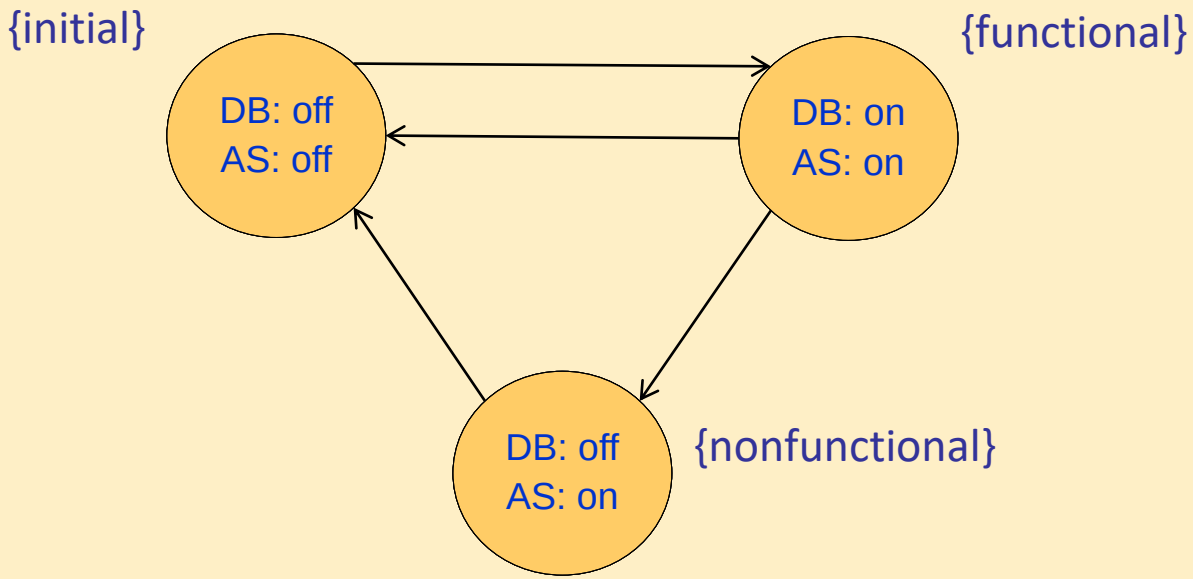
Model checking: Servers

- An IT system has two servers, a database server (DB) and an application server (AS), both of which can be turned on or off.
- Initially, both servers are off. During normal operation, the servers are turned on and off simultaneously.
- The system is functional if both servers are on.
- If – in the functional state – the database server is turned off due to an error, then the system becomes nonfunctional. After this, the application server is also turned off, then the system is restarted by turning both servers on again.
- Tasks:
  1. Create a Kripke structure modeling the behavior of the system described above with regard to the states of the servers! Label the states with the following atomic propositions (based on the informal description):
     {initial, functional, nonfunctional}
  2. Check if the following CTL formula holds for the functional state of the Kripke structure:
     $E(\neg \text{nonfunctional } U \text{ initial})$

Model checking: Servers – Solution 1/2

- An IT system has two servers, a database server (DB) and an application server (AS), both of which can be turned on or off.
- Initially, both servers are off. During normal operation, the servers are turned on and off simultaneously.
- The system is functional if both servers are on.
- If – in the functional state – the database server is turned off due to an error, then the system becomes nonfunctional. After this, the application server is also turned off, then the system is restarted by turning both servers on again.

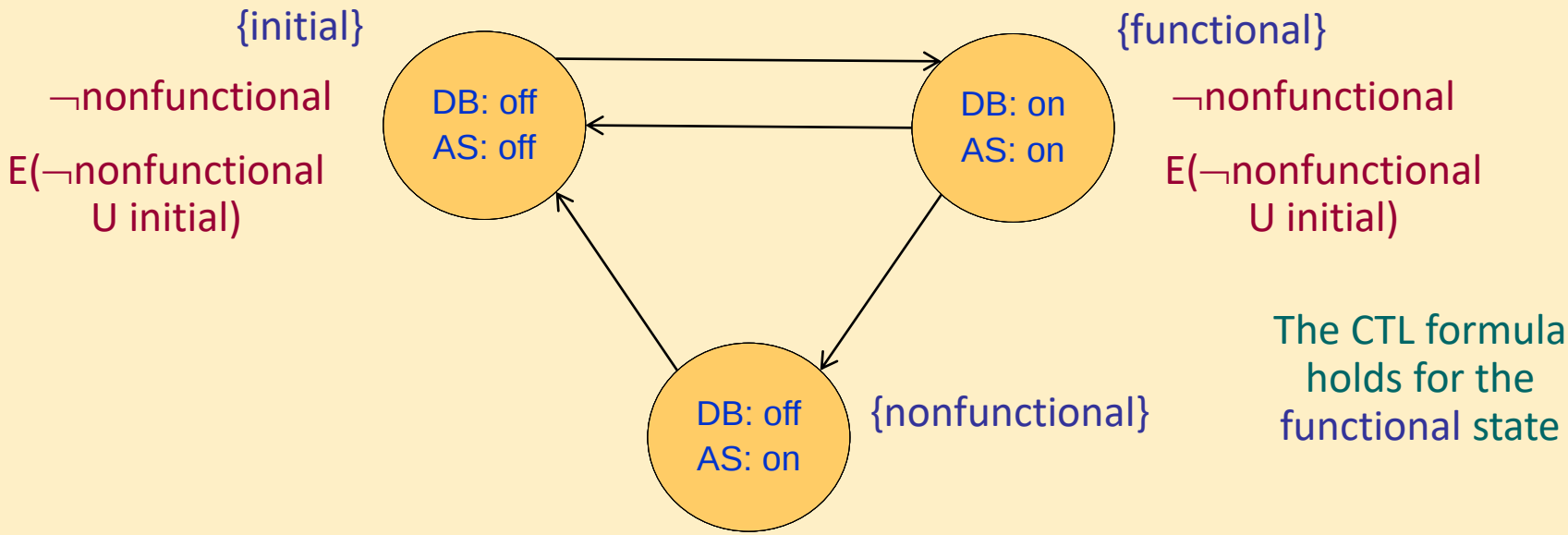
Kripke structure of the system:



Model checking: Servers – Solution 2/2

- An IT system has two servers, a database server (DB) and an application server (AS), both of which can be turned on or off.
- Initially, both servers are off. During normal operation, the servers are turned on and off simultaneously.
- The system is functional if both servers are on.
- If – in the functional state – the database server is turned off due to an error, then the system becomes nonfunctional. After this, the application server is also turned off, then the system is restarted by turning both servers on again.

Labeling the Kripke structure with the properties:



ROBDD: Building ROBDD

Consider the following Boolean function *g*:

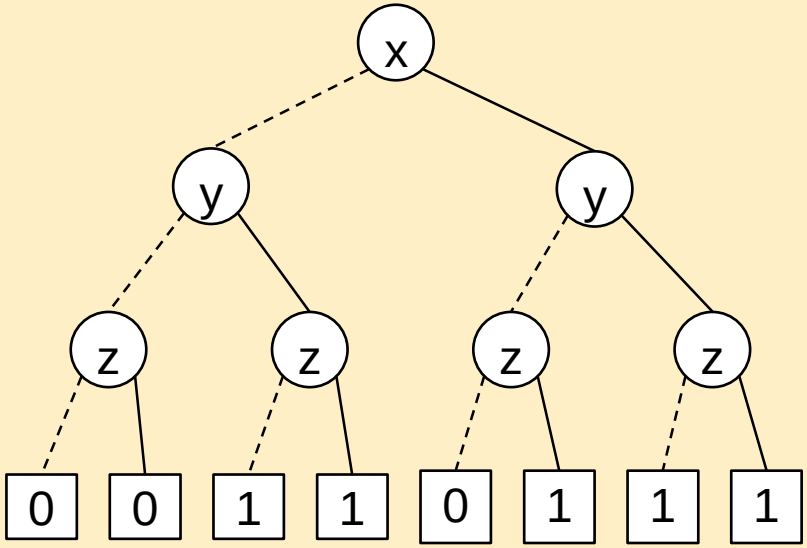
x	y	z	f(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

1. Construct the decision tree representing *g*! Use the variable ordering used in the table: *x*, *y*, *z*.
2. Based on this, construct the reduced ordered binary decision diagram (ROBDD) representation of *g*!
3. Give the algebraic form of the function!

ROBDD: Building ROBDD – Solution 1/3

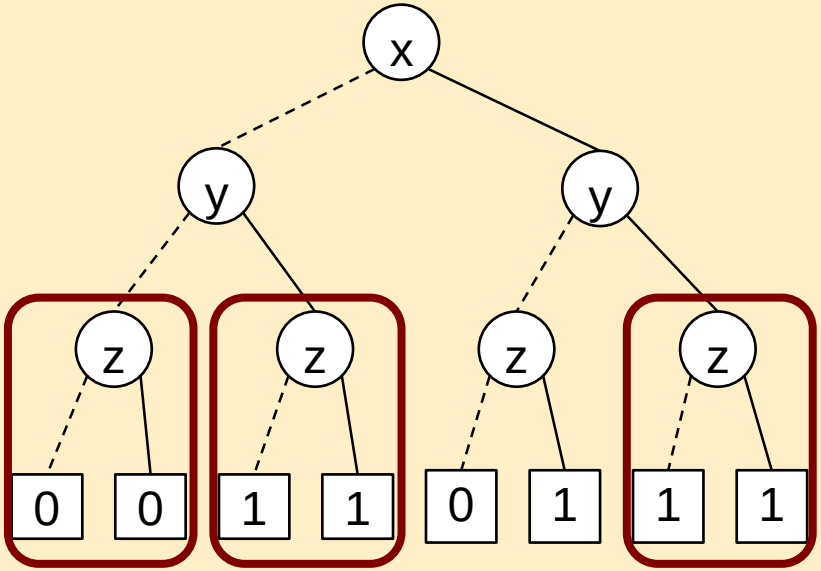
Constructing the decision tree for function *g*:

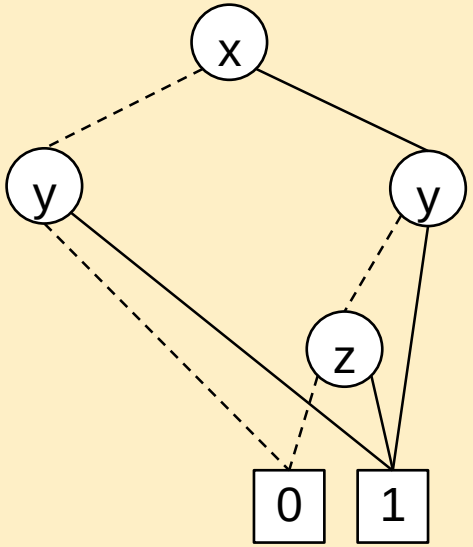
x	y	z	f(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



ROBDD: Building ROBDD – Solution 2/3

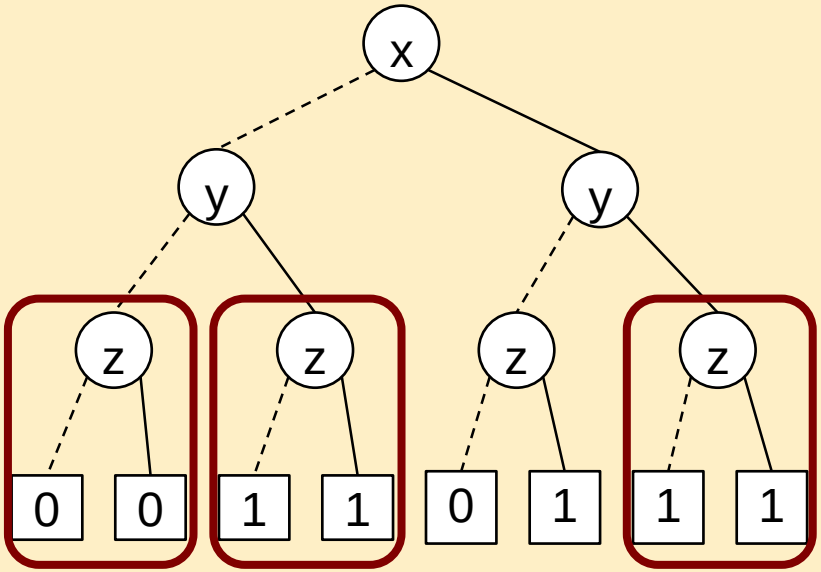
Constructing the ROBDD for function *g*:

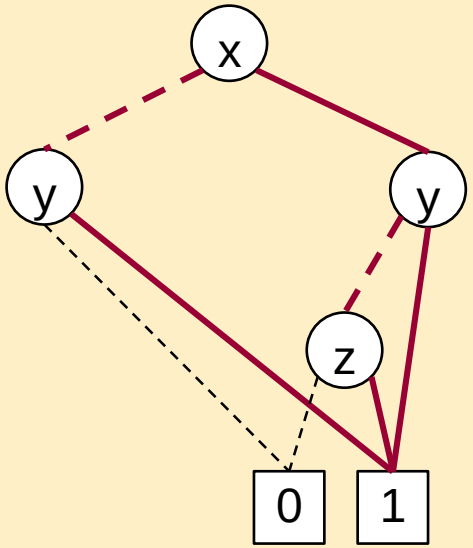




ROBDD: Building ROBDD – Solution 3/3

Constructing the ROBDD for function *g*:



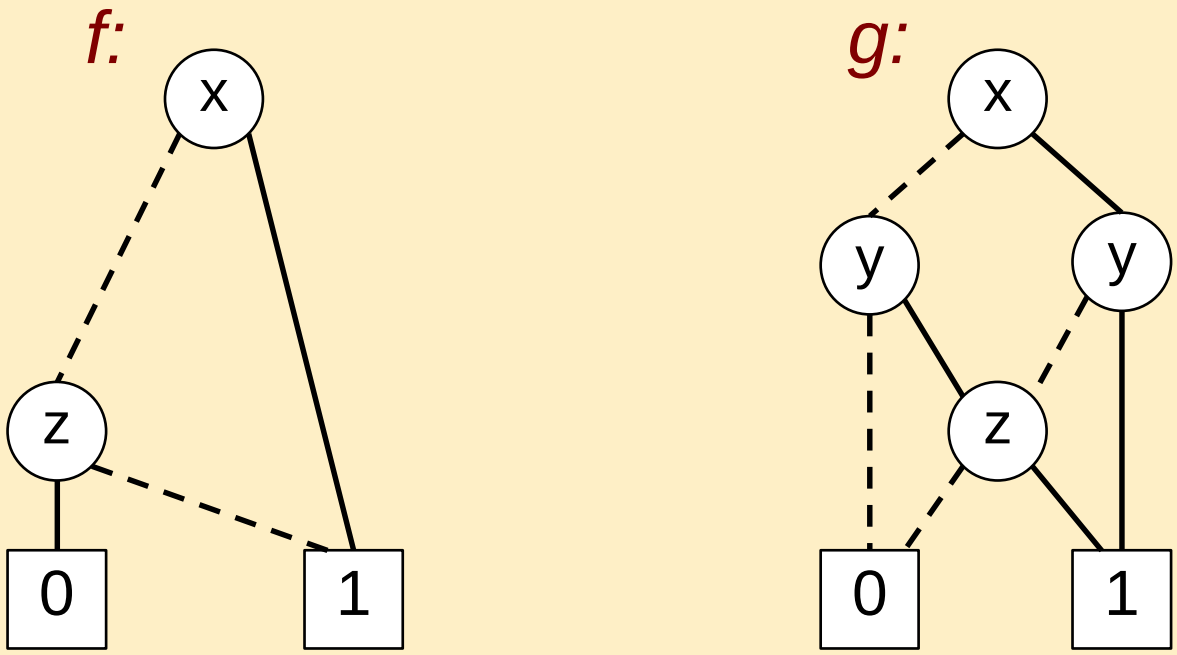


Algebraic function: Following the paths to 1:

$$g = (\neg x \wedge y) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y)$$

ROBDD: Operations on functions

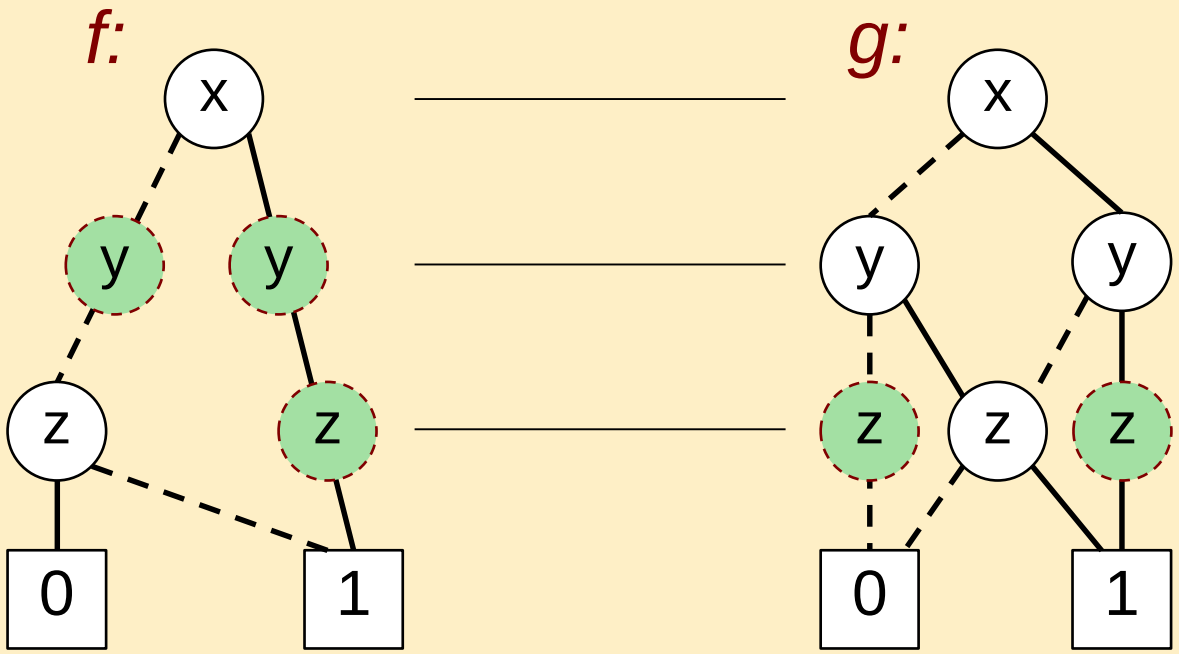
Consider the following functions f and g given in ROBDD form. Construct the ROBDD representing $f \wedge g$!



ROBDD: Operations on functions

Consider the following functions f and g given in ROBDD form. Construct the ROBDD representing $f \wedge g$!

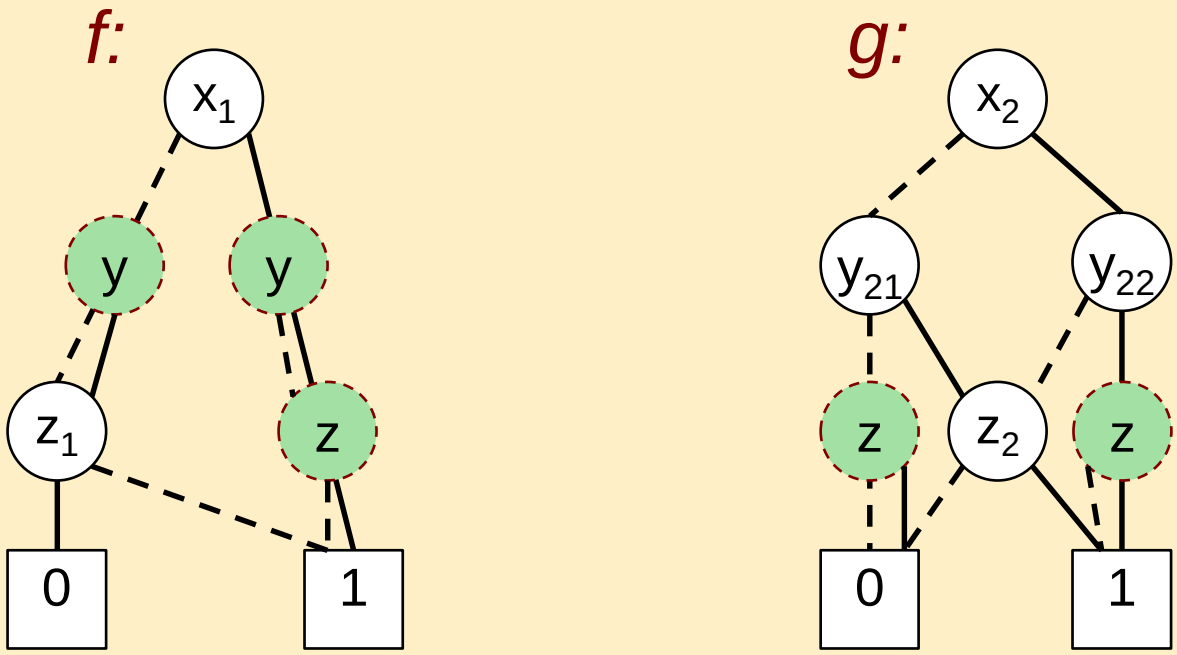
Missing (reduced) nodes:



ROBDD: Operations on functions

Consider the following functions f and g given in ROBDD form. Construct the ROBDD representing $f \wedge g$!

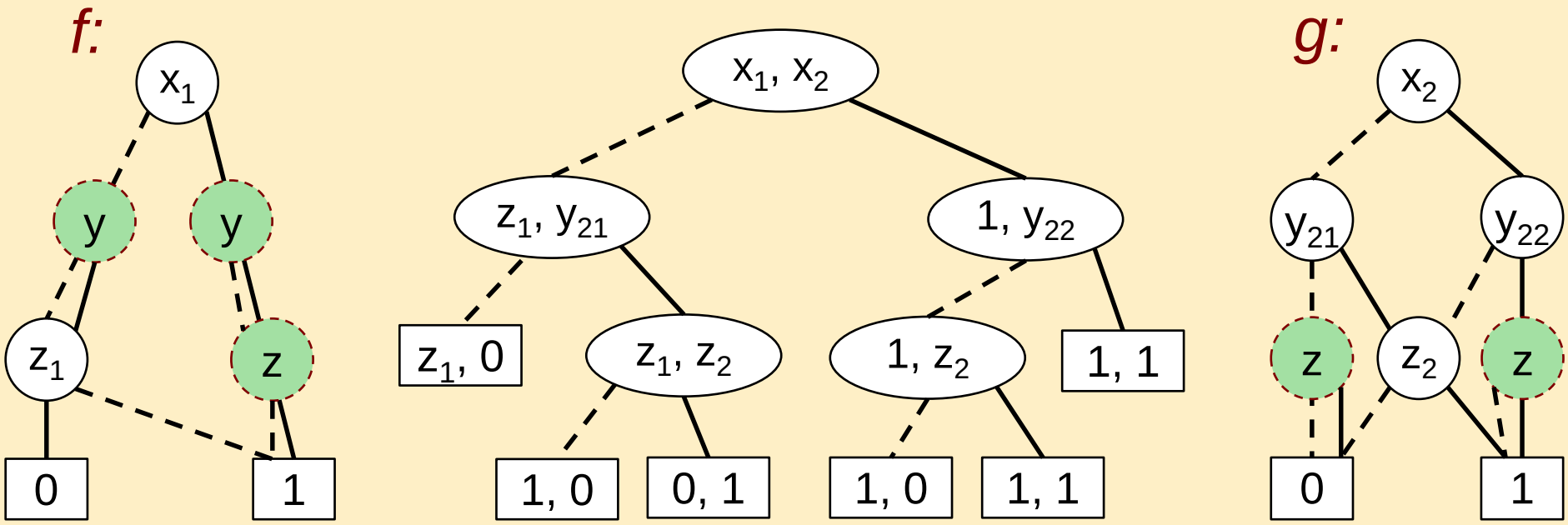
Identifying the existing nodes:



ROBDD: Operations on functions – Solution

Constructing the ROBDD representing $f \wedge g$!

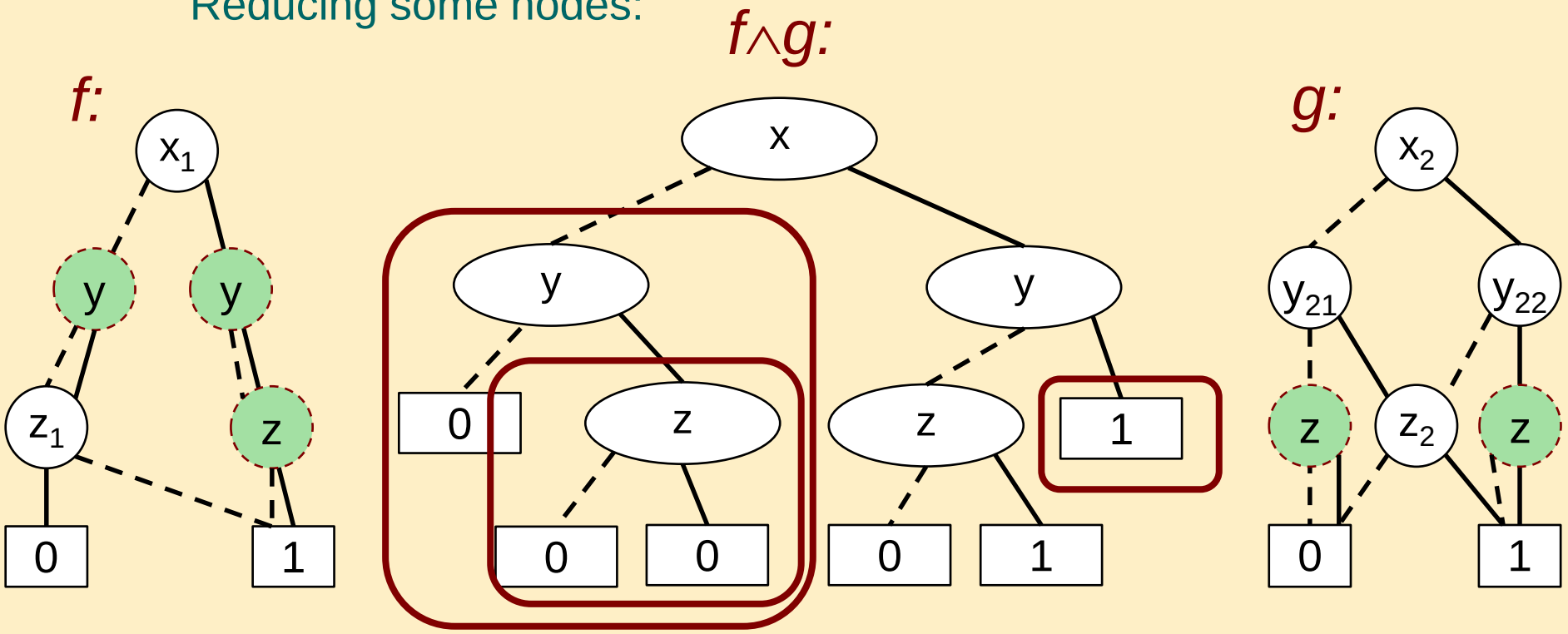
"Combining" the nodes:



ROBDD: Operations on functions – Solution

Constructing the ROBDD representing $f \wedge g$!

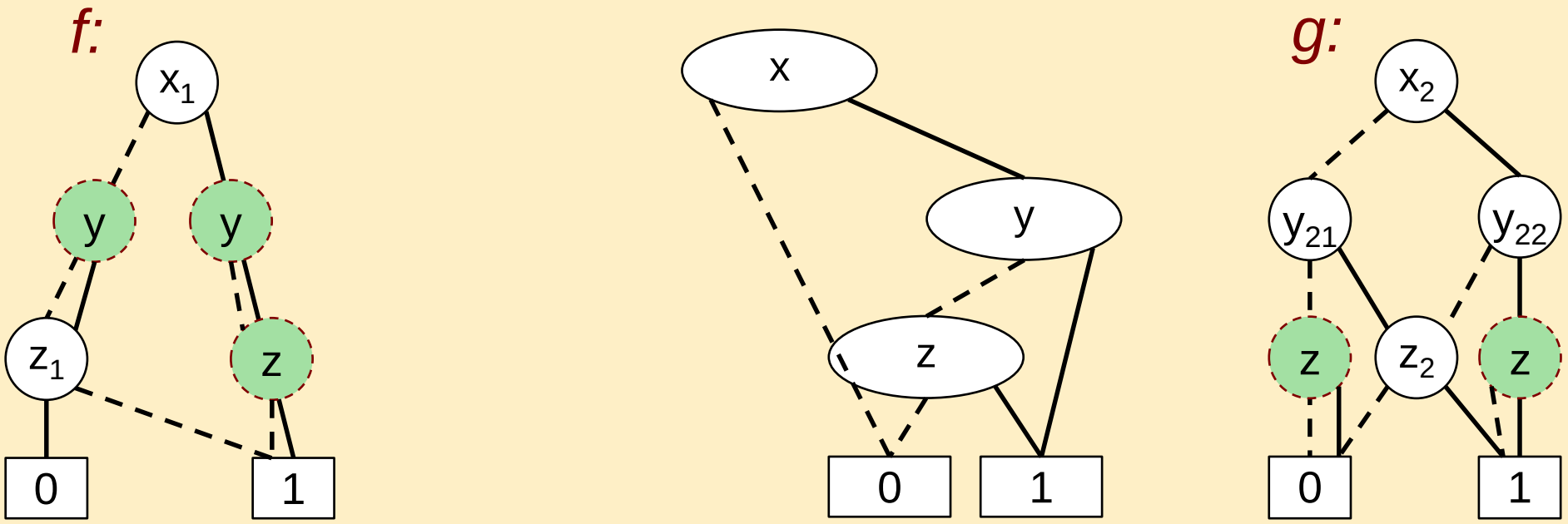
Reducing some nodes:



ROBDD: Operations on functions – Solution

Constructing the ROBDD representing $f \wedge g$!

Reducing some nodes:



ROBDD: Operations on functions – Solution

Constructing the ROBDD representing $f \wedge g$!

The resulting ROBDD: