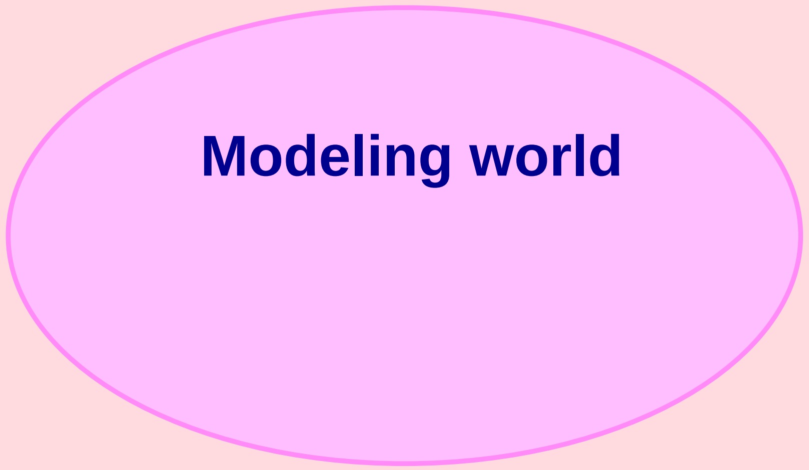
CTL May Be Ambiguous when Model Checking Moore Machines

Cédric Roux and Emmanuelle Encrenaz
Université Pierre et Marie Curie
Laboratoire d’Informatique de Paris 6
Architecture des Systèmes Intégrés et Micro−électronique

Modeling versus Verification

Modeling versus Verification



Modeling versus Verification

Modeling world

Moore or Mealy machines

Modeling versus Verification

Verification world

Modeling world

Moore or Mealy machines

Modeling versus Verification

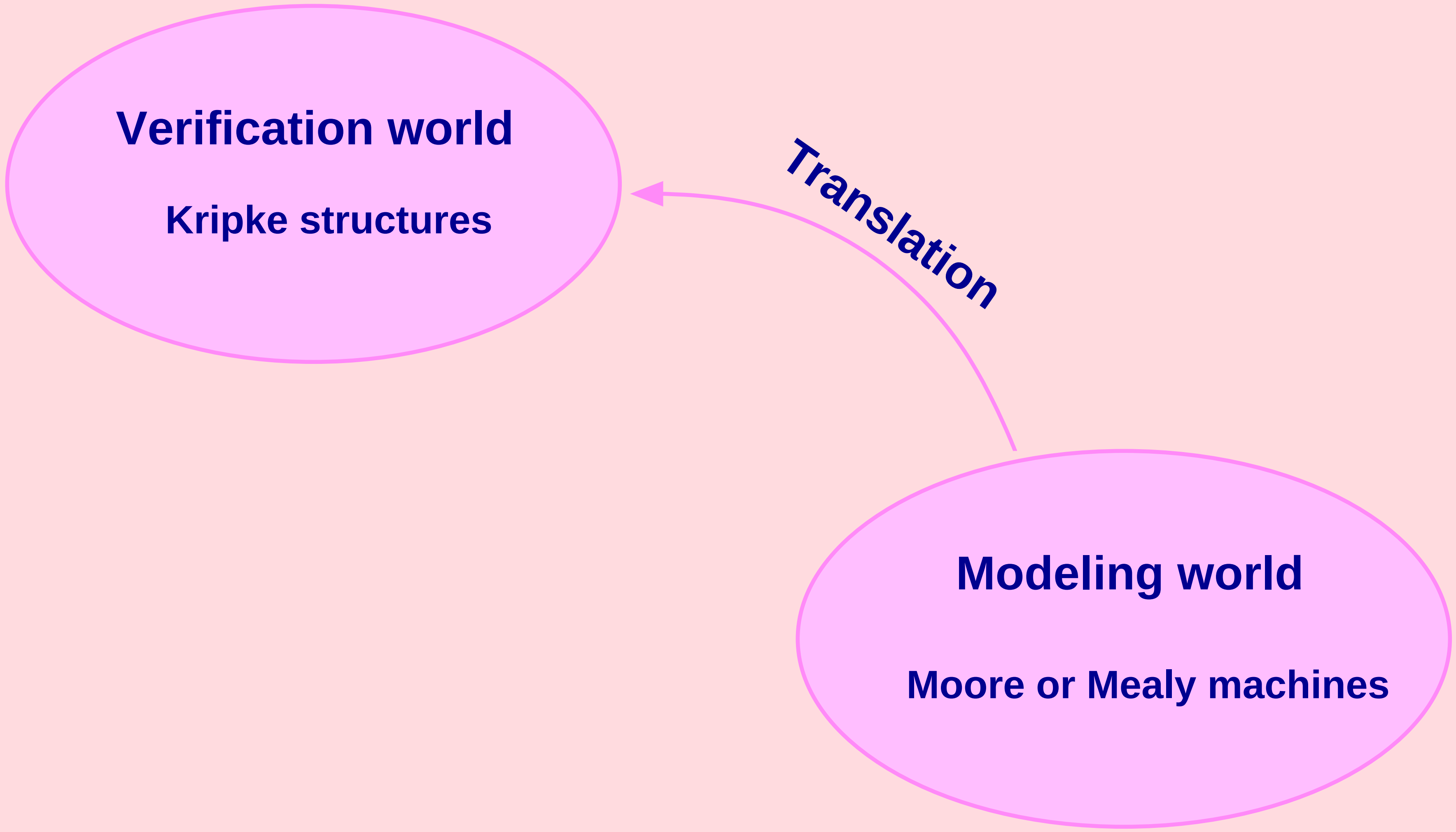
Verification world

Kripke structures

Modeling world

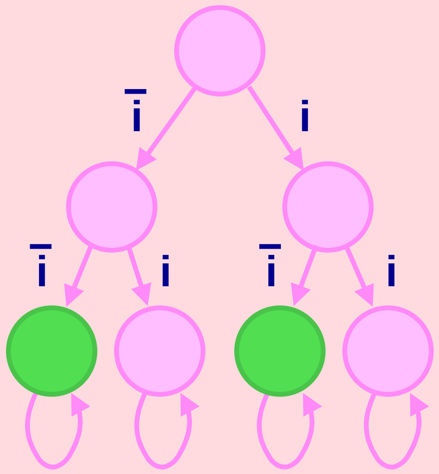
Moore or Mealy machines

Modeling versus Verification



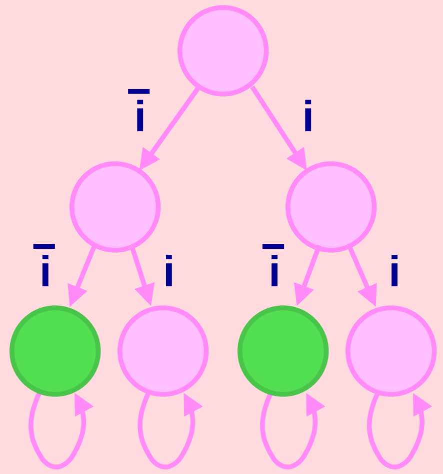
From Moore to Kripke

From Moore to Kripke



From Moore to Kripke

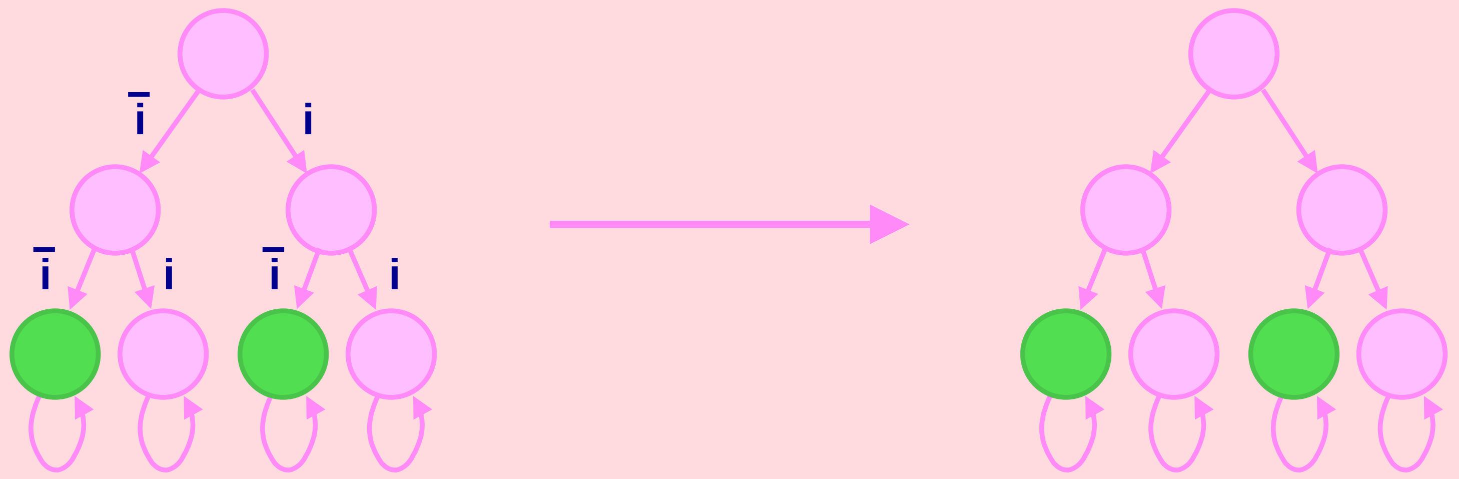
First translation scheme



From Moore to Kripke

First translation scheme

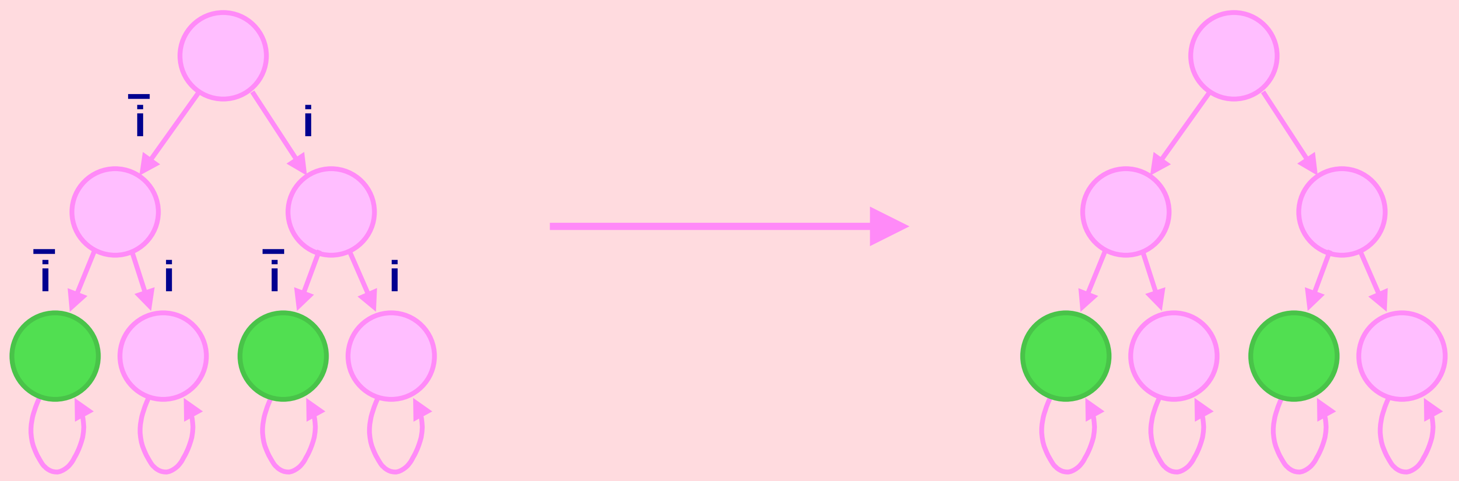
Remove the input signals



From Moore to Kripke

First translation scheme

Remove the input signals

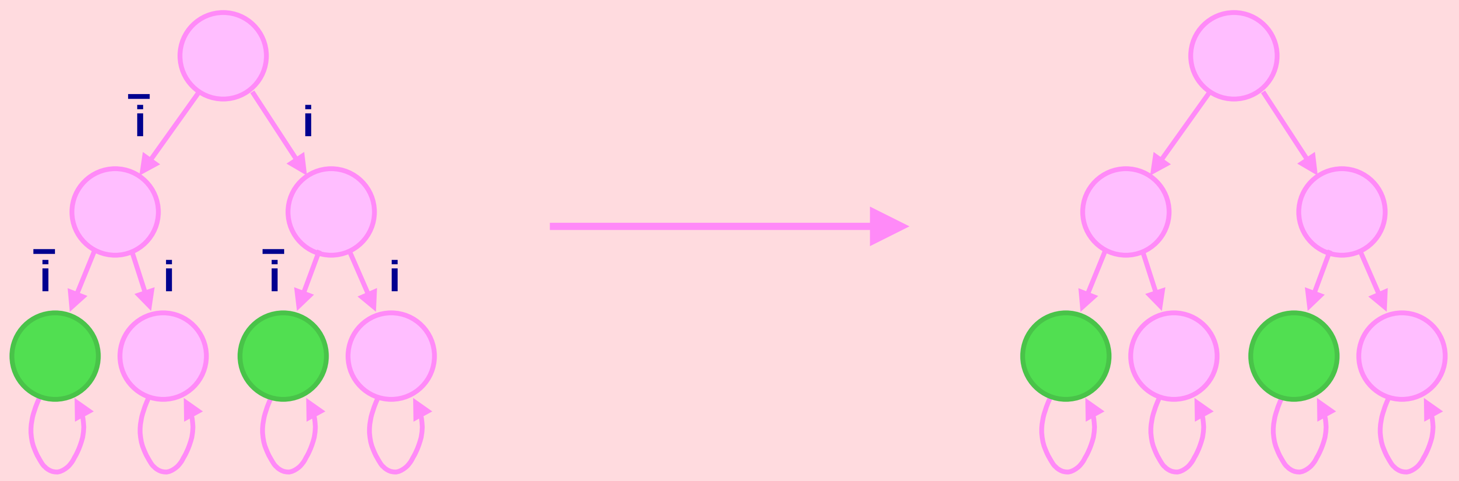


Simple

From Moore to Kripke

First translation scheme

Remove the input signals

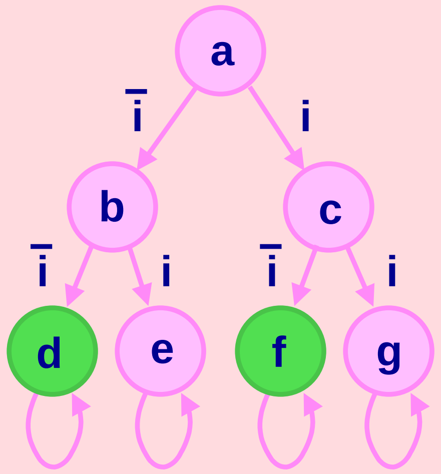


Simple

Impossible to express properties including input signals

From Moore to Kripke

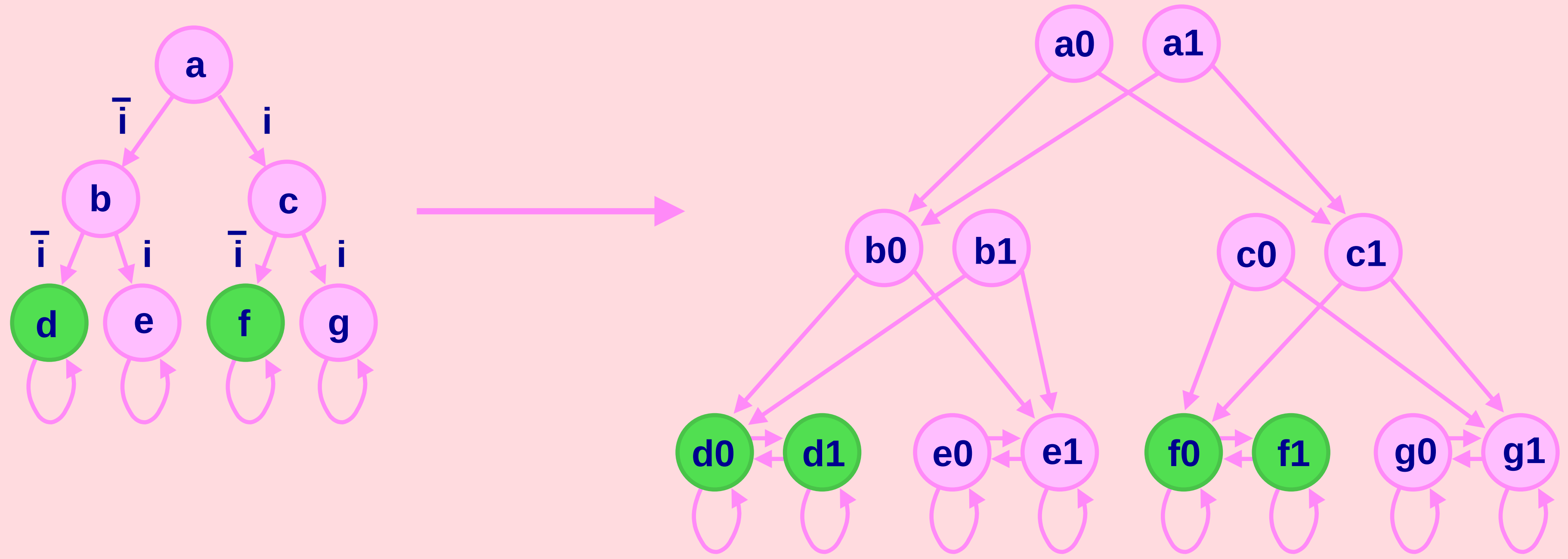
Second translation scheme



From Moore to Kripke

Second translation scheme

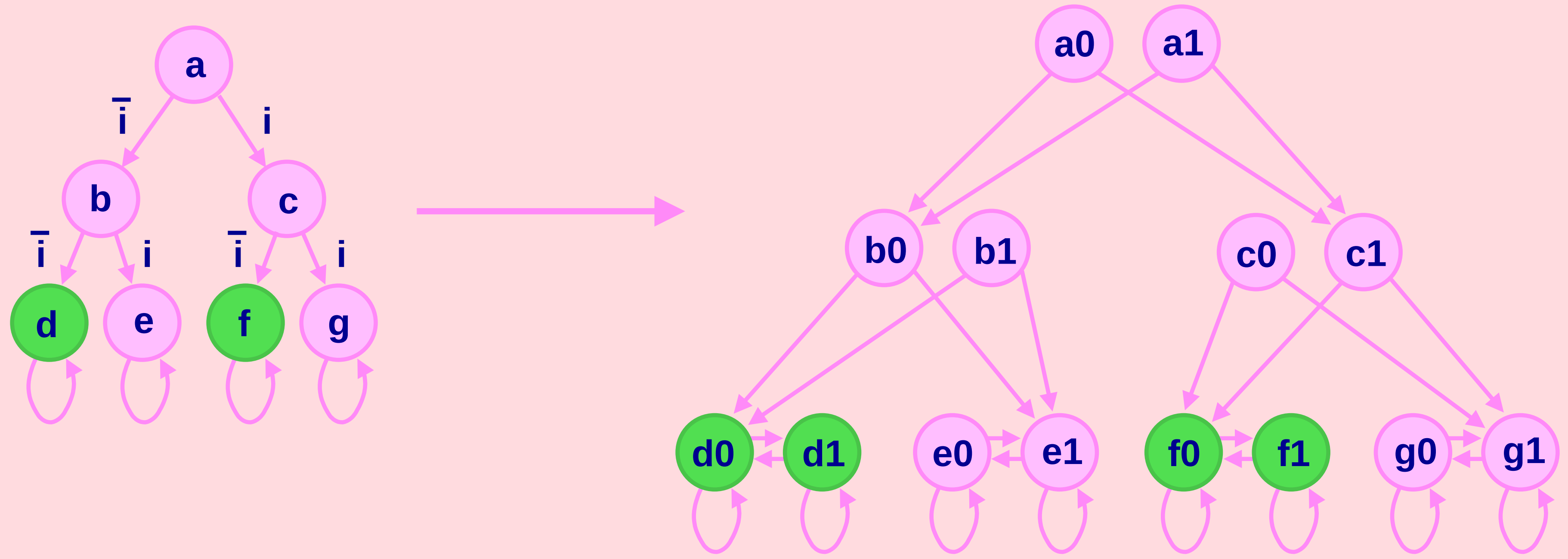
Input signals into target state of transitions



From Moore to Kripke

Second translation scheme

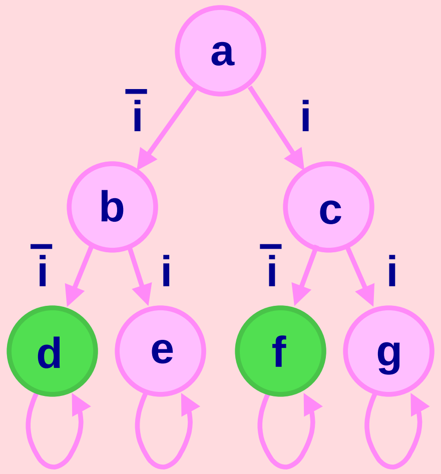
Input signals into target state of transitions



Composition of Moore machines lost

From Moore to Kripke

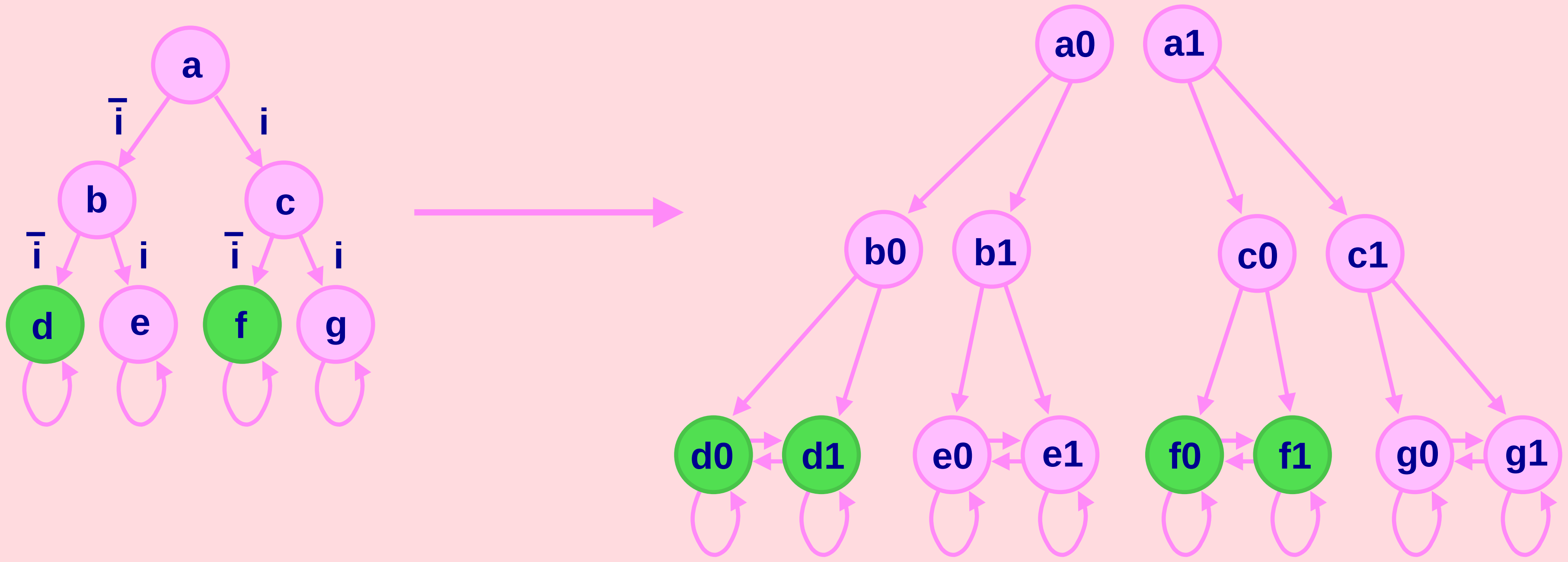
Third translation scheme



From Moore to Kripke

Third translation scheme

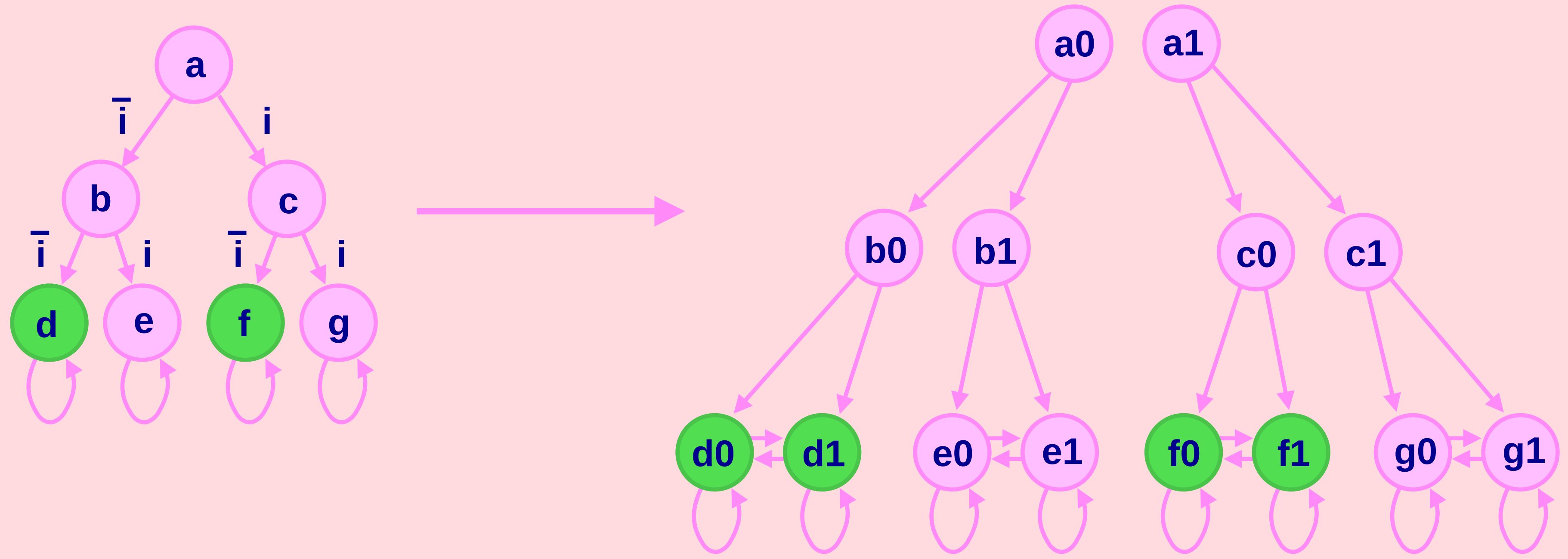
Input signals into source state of transitions



From Moore to Kripke

Third translation scheme

Input signals into source state of transitions

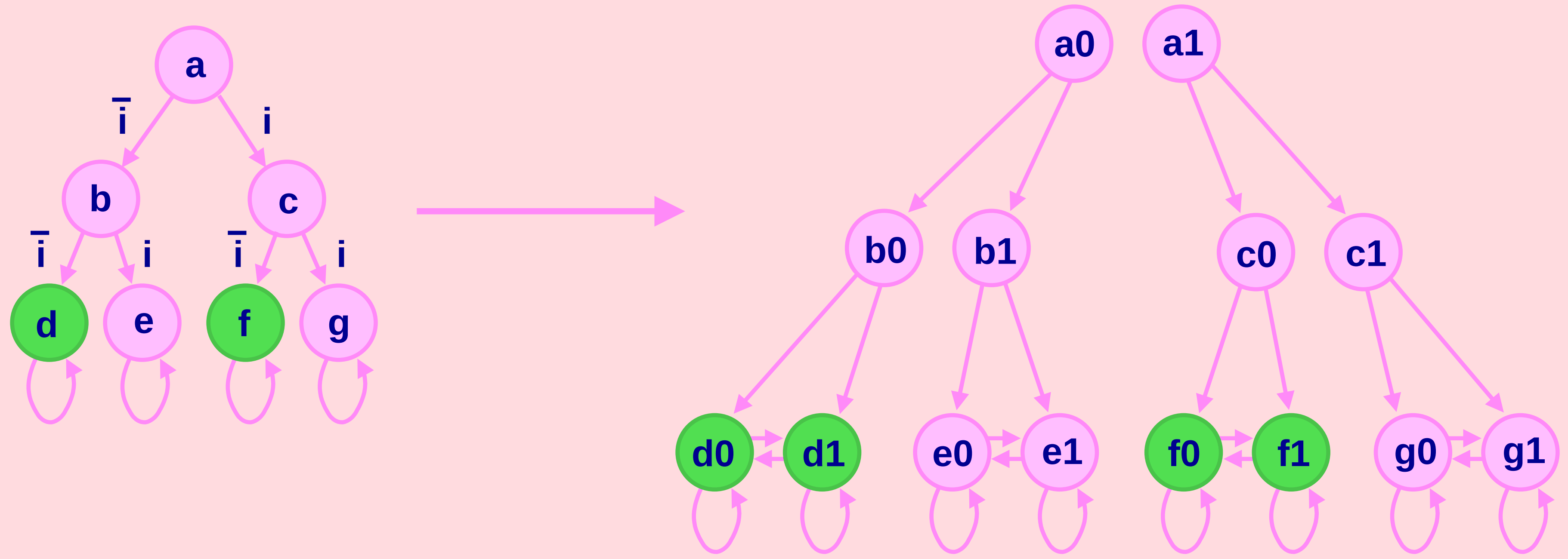


We can compose Moore machines

From Moore to Kripke

Third translation scheme

Input signals into source state of transitions



We can compose Moore machines

This may introduce ambiguities when using CTL

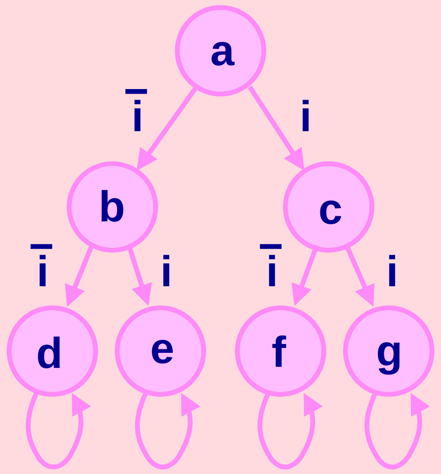
Possible CTL ambiguities

Possible CTL ambiguities

Checking the property AX EX p

Possible CTL ambiguities

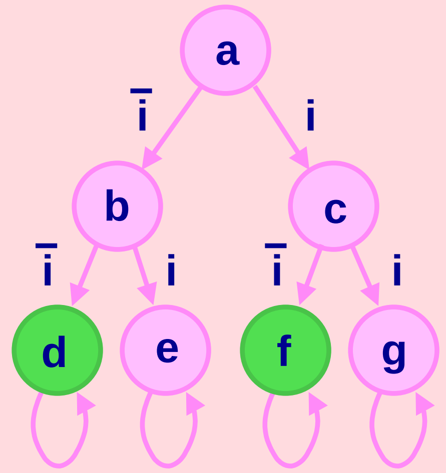
Checking the property AX EX p



Possible CTL ambiguities

Checking the property AX EX p

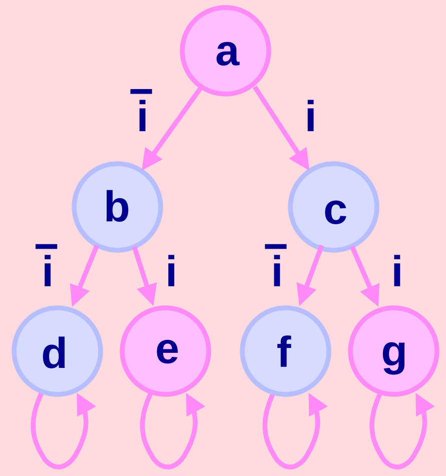
🟢 states verifying p



Possible CTL ambiguities

Checking the property AX EX p

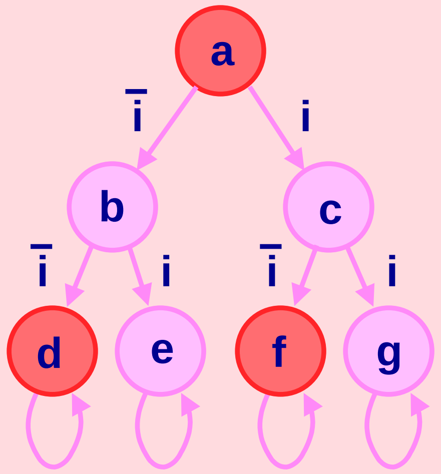
● states verifying EX p



Possible CTL ambiguities

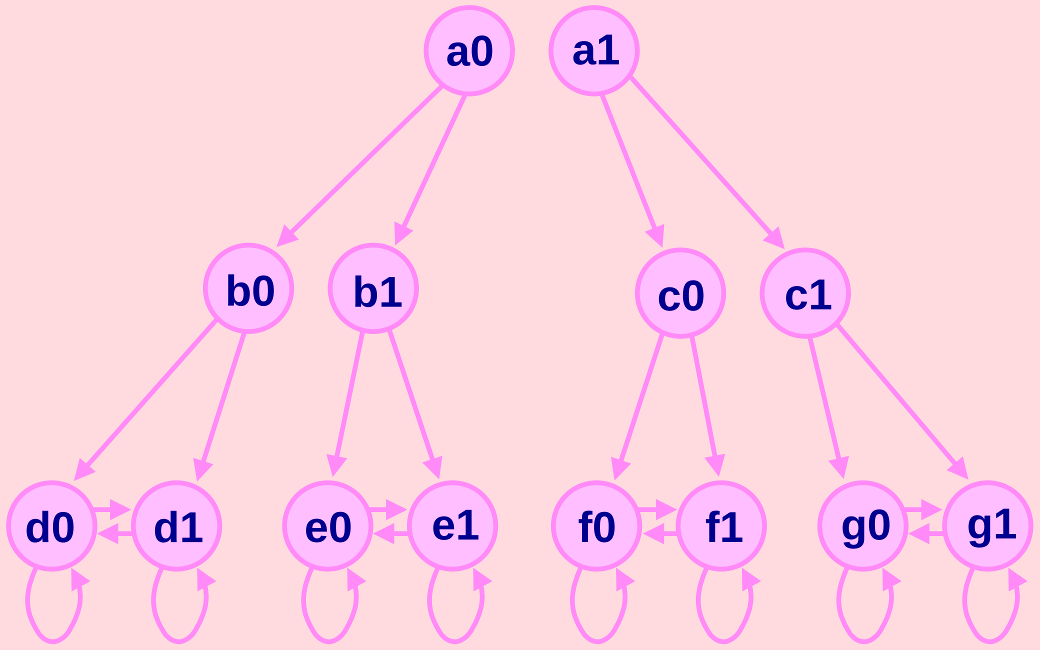
Checking the property AX EX p

🔴 states verifying AX EX p



Possible CTL ambiguities

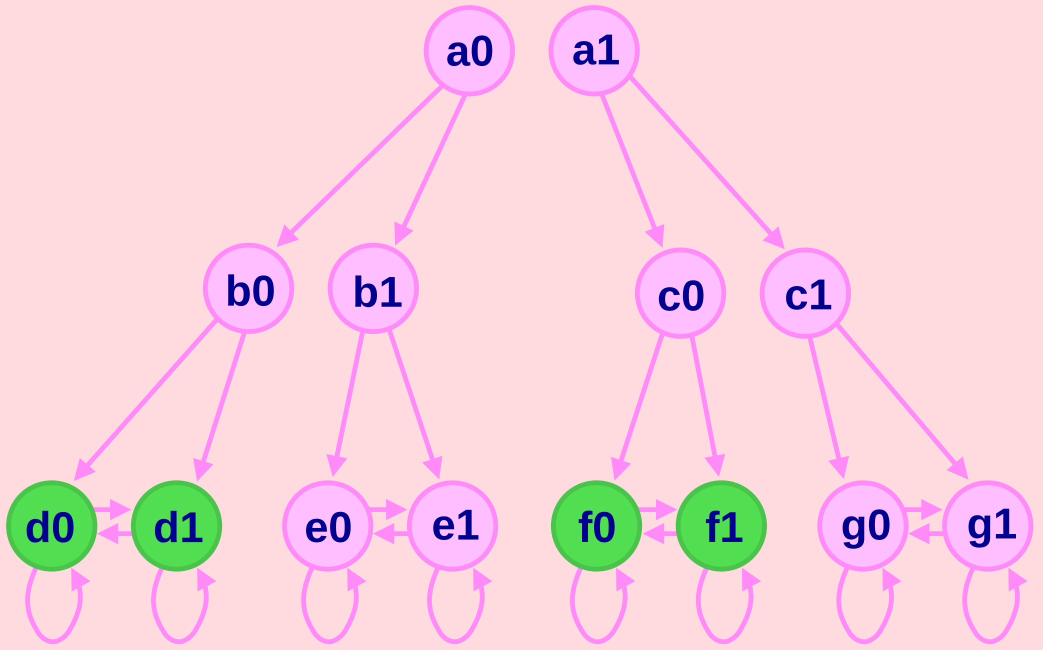
Checking the property AX EX p



Possible CTL ambiguities

Checking the property AX EX p

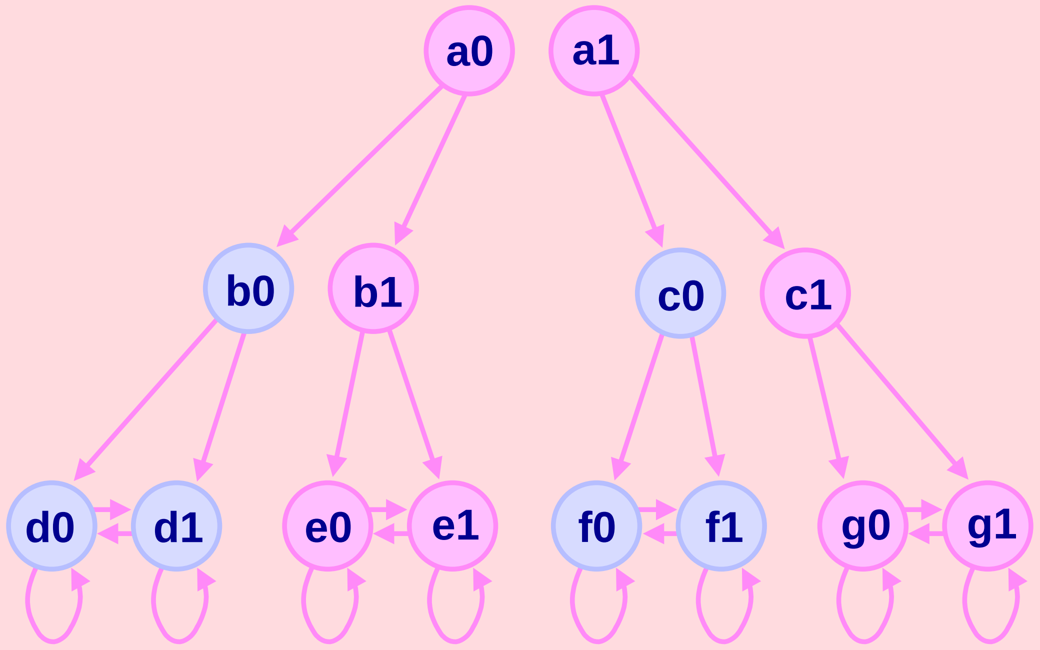
● states verifying p



Possible CTL ambiguities

Checking the property AX EX p

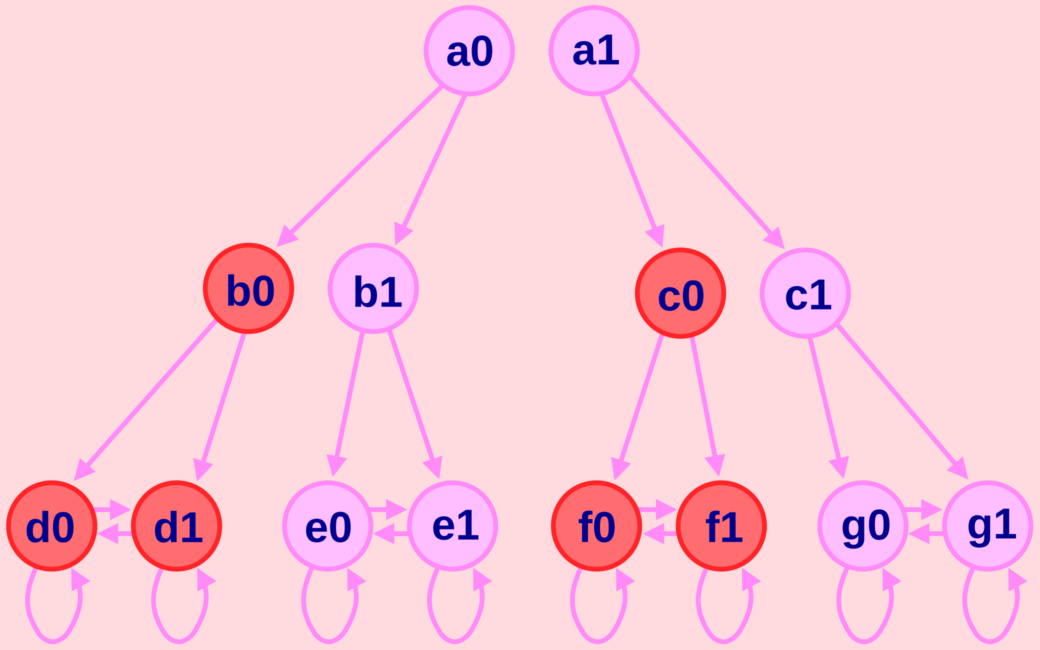
● states verifying EX p



Possible CTL ambiguities

Checking the property AX EX p

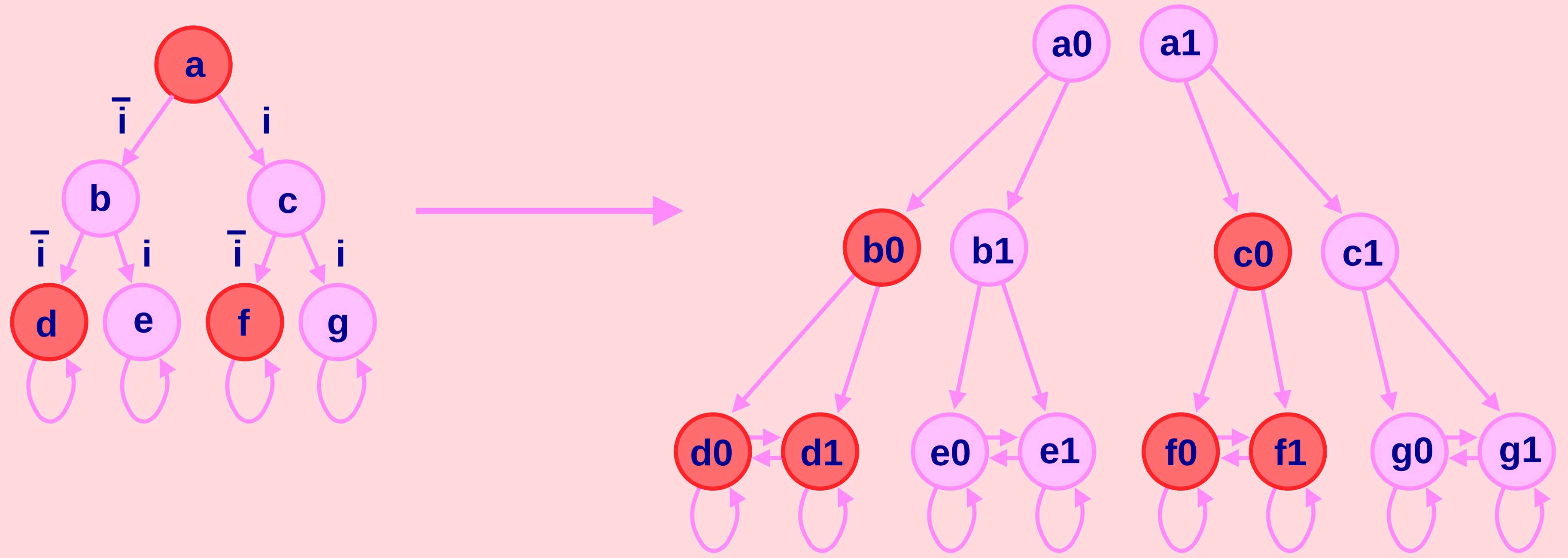
● states verifying AX EX p



Possible CTL ambiguities

Checking the property AX EX p

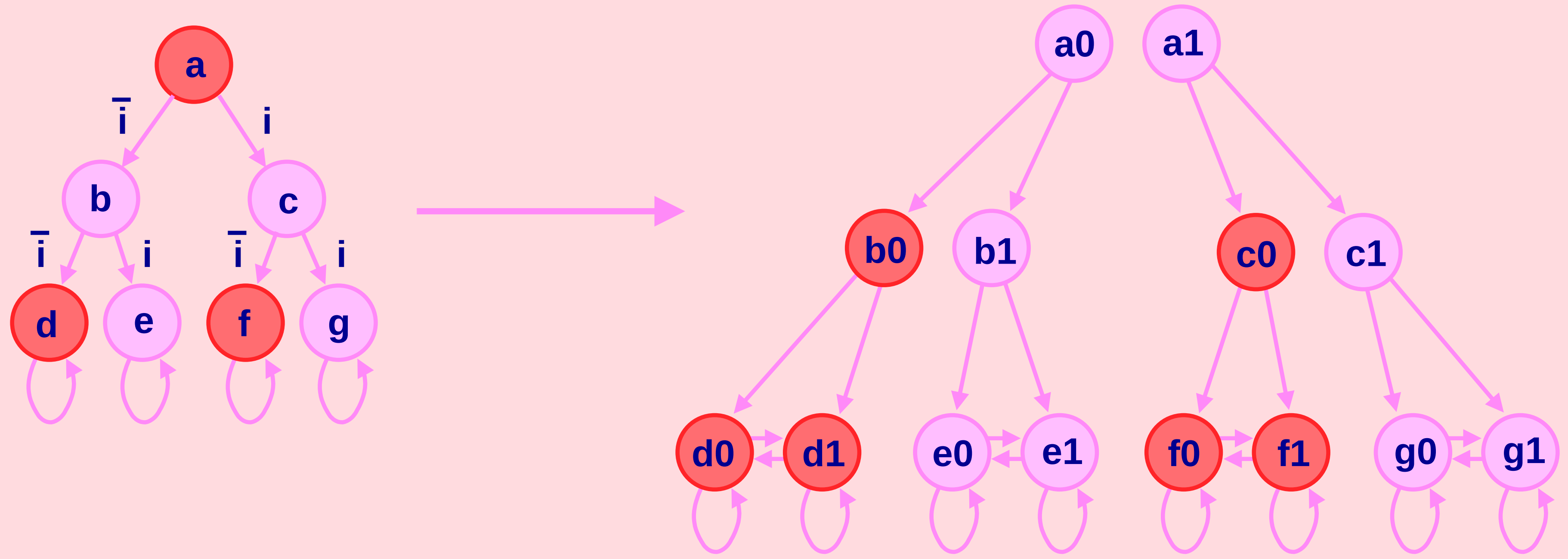
● states verifying AX EX p



Possible CTL ambiguities

Checking the property AX EX p

● states verifying AX EX p



«AX EX p does not have the same truth value in both structures»

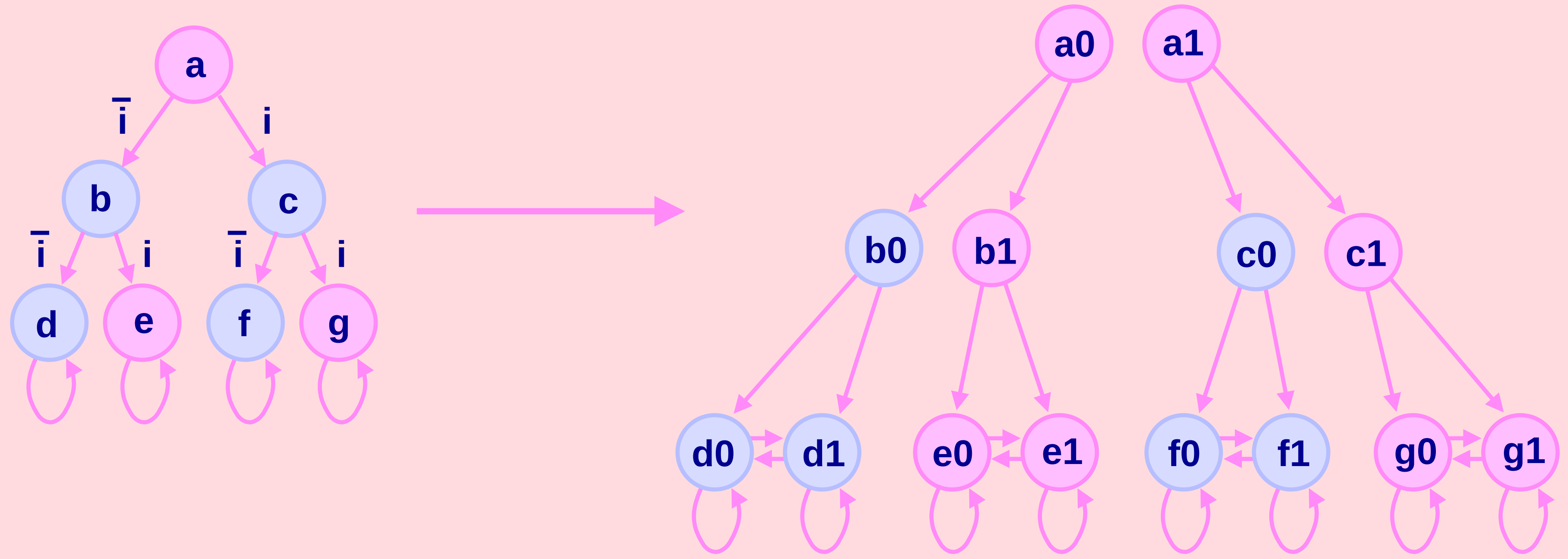
Possible CTL ambiguities

A first ambiguity

Possible CTL ambiguities

A first ambiguity

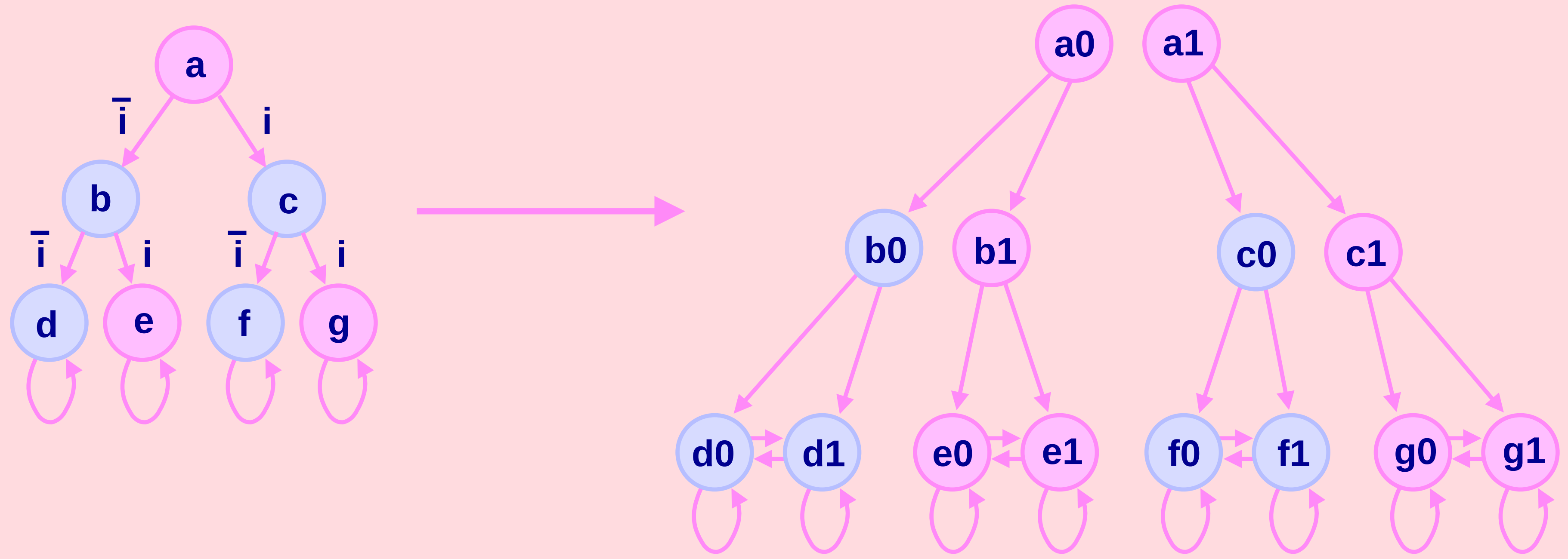
● states verifying EX p



Possible CTL ambiguities

A first ambiguity

● states verifying EX p



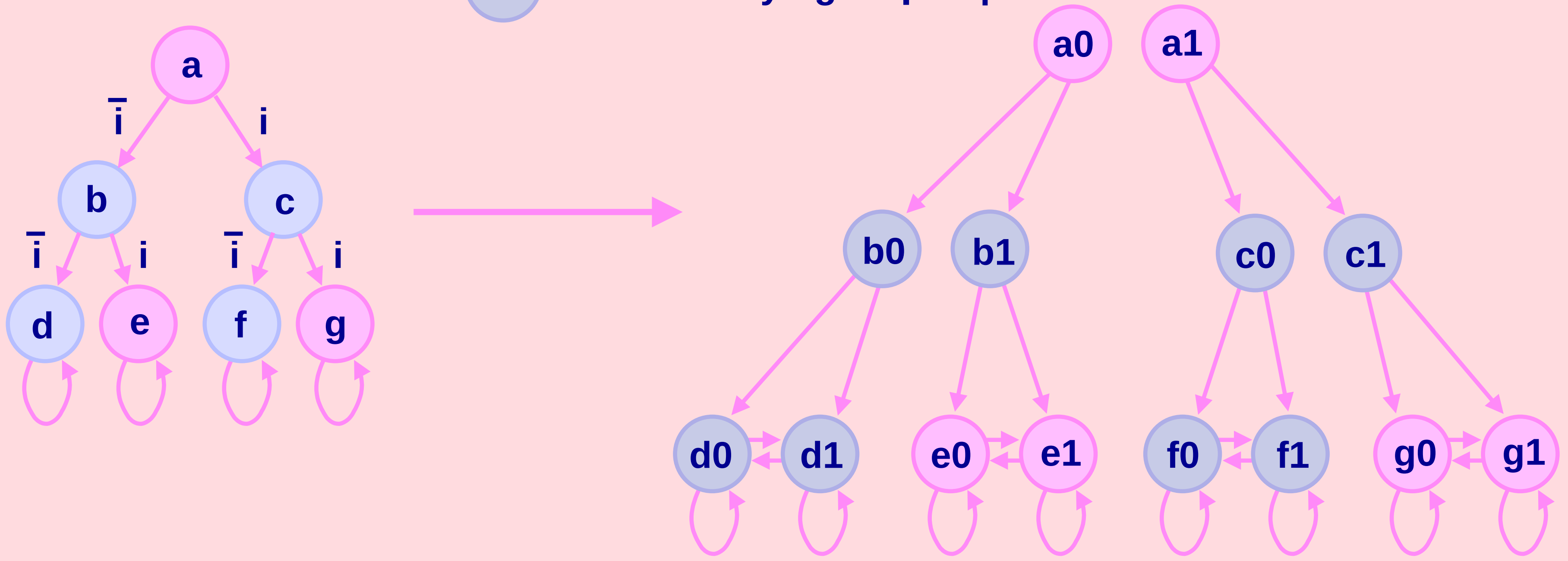
States b0 and b1 should verify EX p, as state b does

Possible CTL ambiguities

A first ambiguity

● states verifying EX p

● states verifying $\exists_i$ EX p



States b0 and b1 should verify EX p, as state b does

We introduce $\exists_i$ to remove this ambiguity

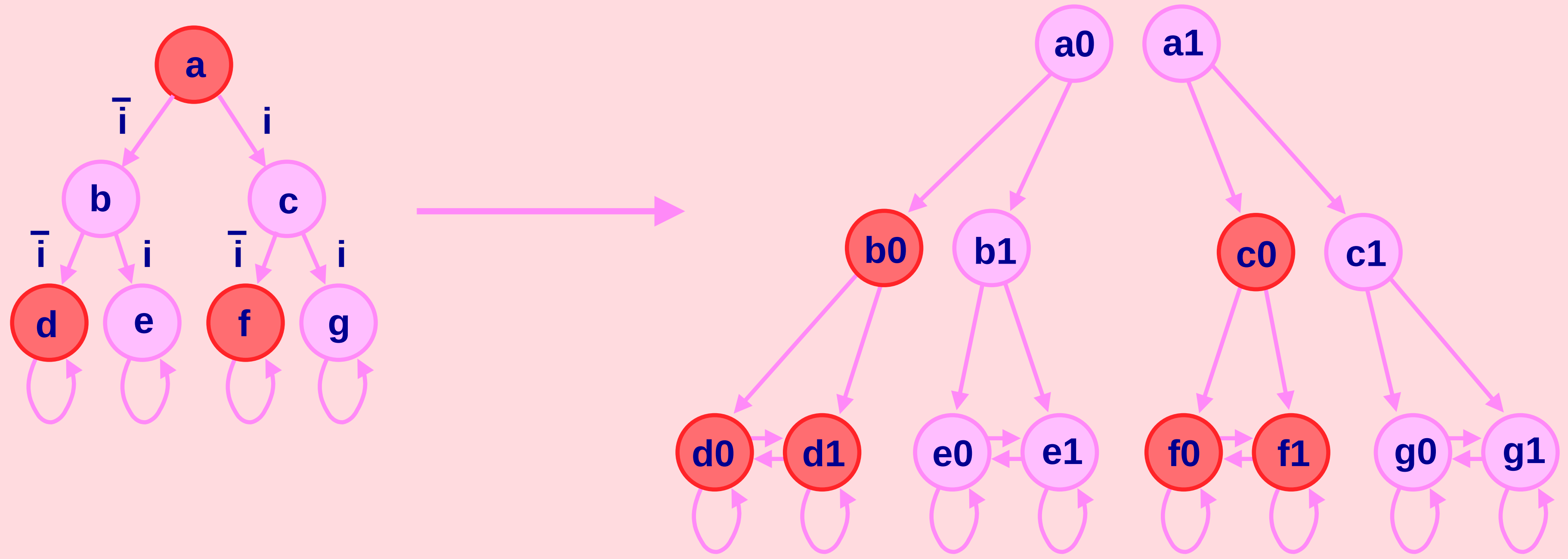
Possible CTL ambiguities

A second ambiguity

Possible CTL ambiguities

A second ambiguity

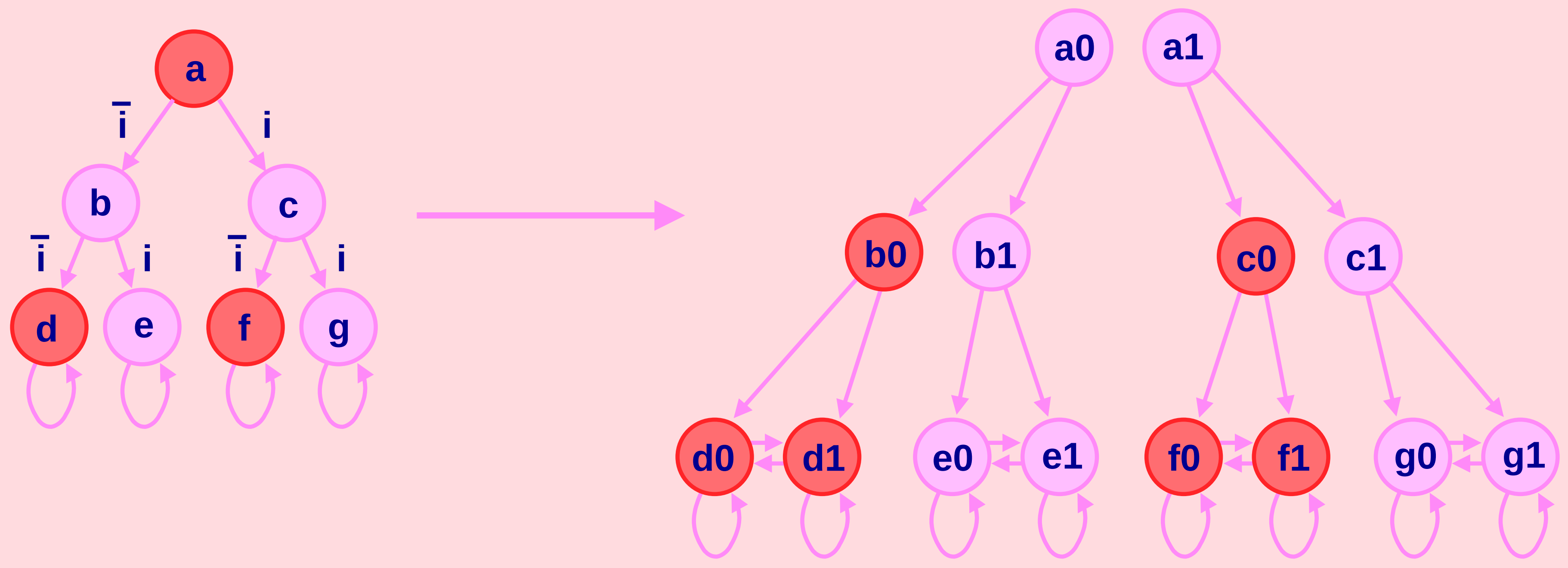
🔴 states verifying AX EX p



Possible CTL ambiguities

A second ambiguity

● states verifying AX EX p



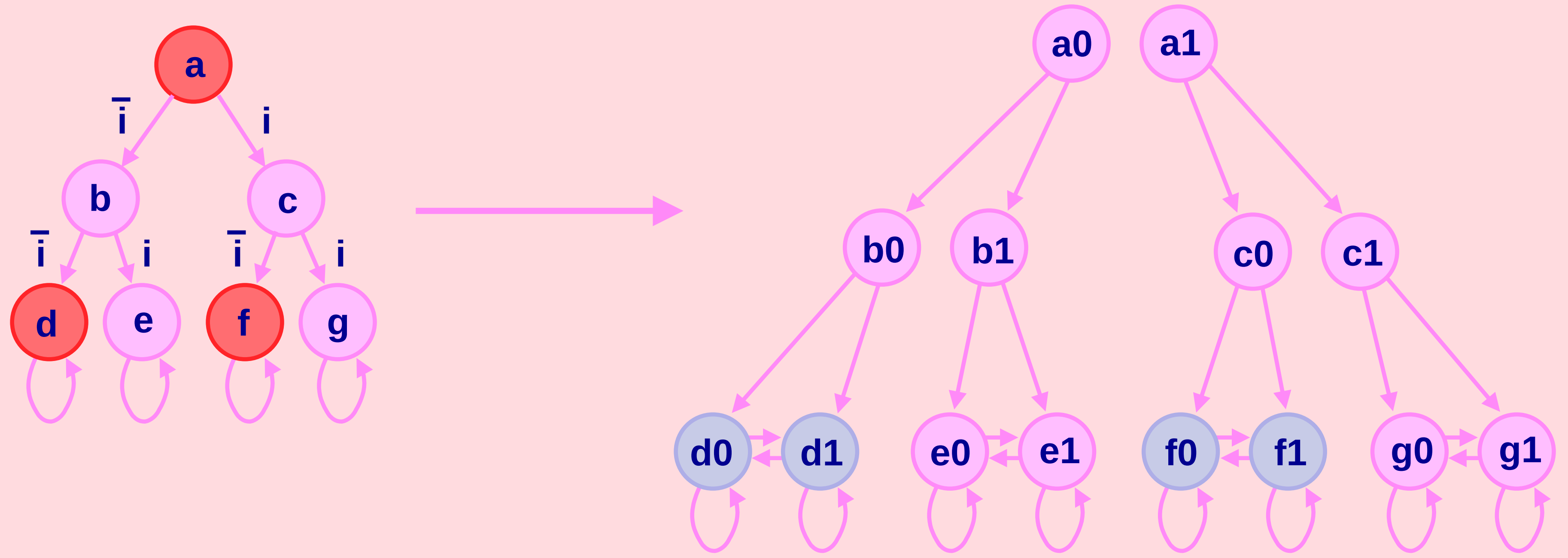
b0 (and b1) should not verify AX EX p, and a0 and a1 should

Possible CTL ambiguities

A second ambiguity

🔴 states verifying AX EX p

⚪ states verifying $\forall_i$ AX EX p



b0 (and b1) should not verify AX EX p, and a0 and a1 should

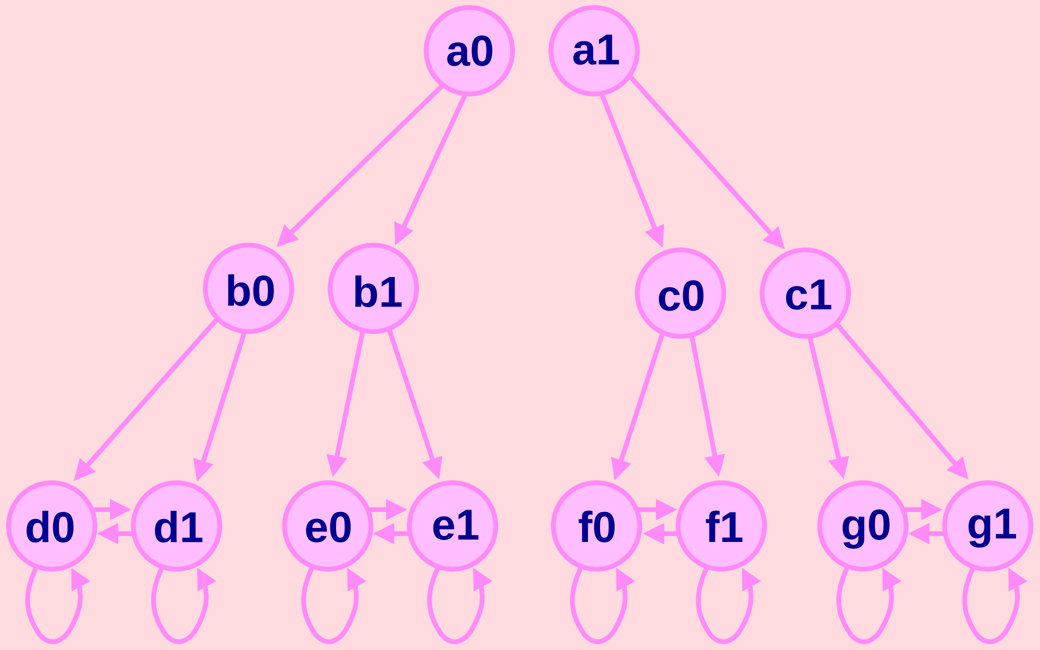
We introduce $\forall_i$ to remove this ambiguity

Possible CTL ambiguities

Checking the property $\forall_i AX \exists_i EX\ p$

Possible CTL ambiguities

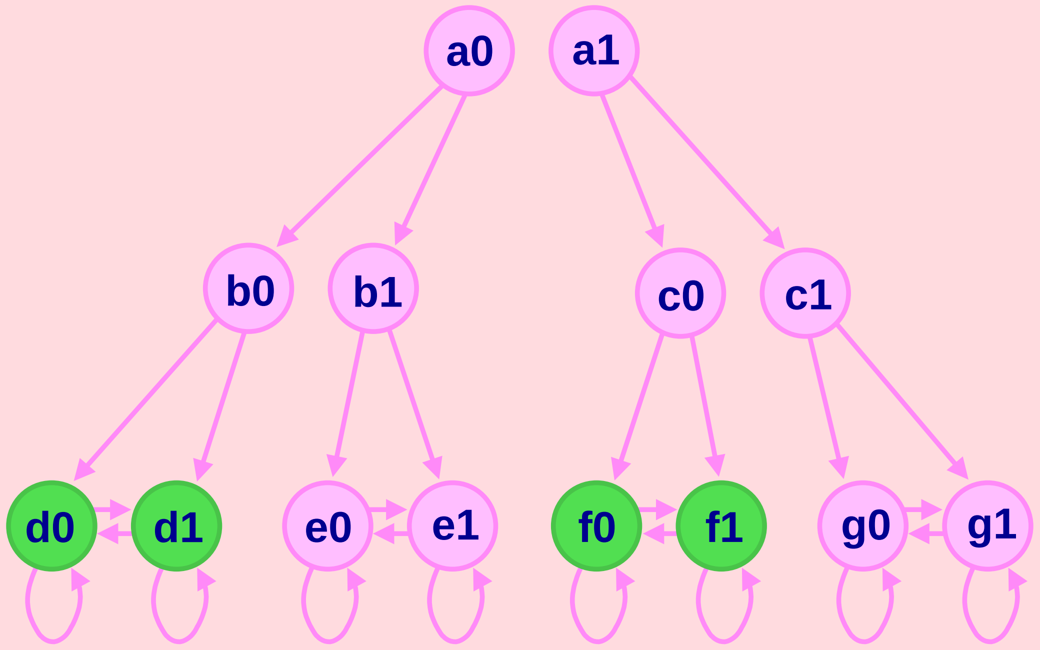
Checking the property $\forall_i \, AX \, \exists_i \, EX \, p$



Possible CTL ambiguities

Checking the property $\forall_i AX \exists_i EX p$

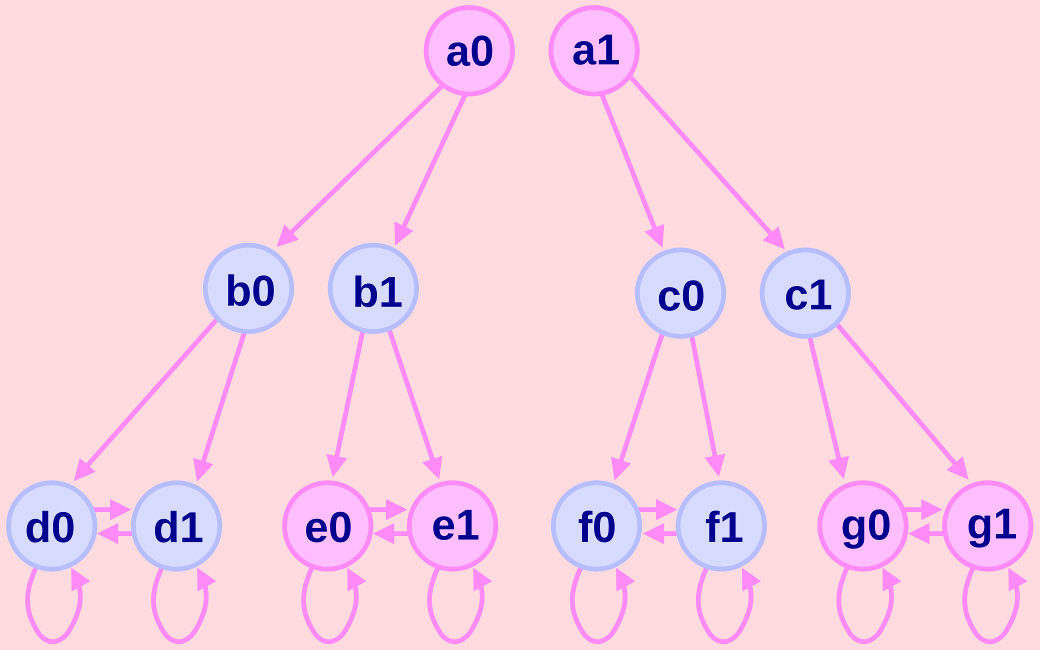
● states verifying p



Possible CTL ambiguities

Checking the property $\forall_i AX \exists_i EX\, p$

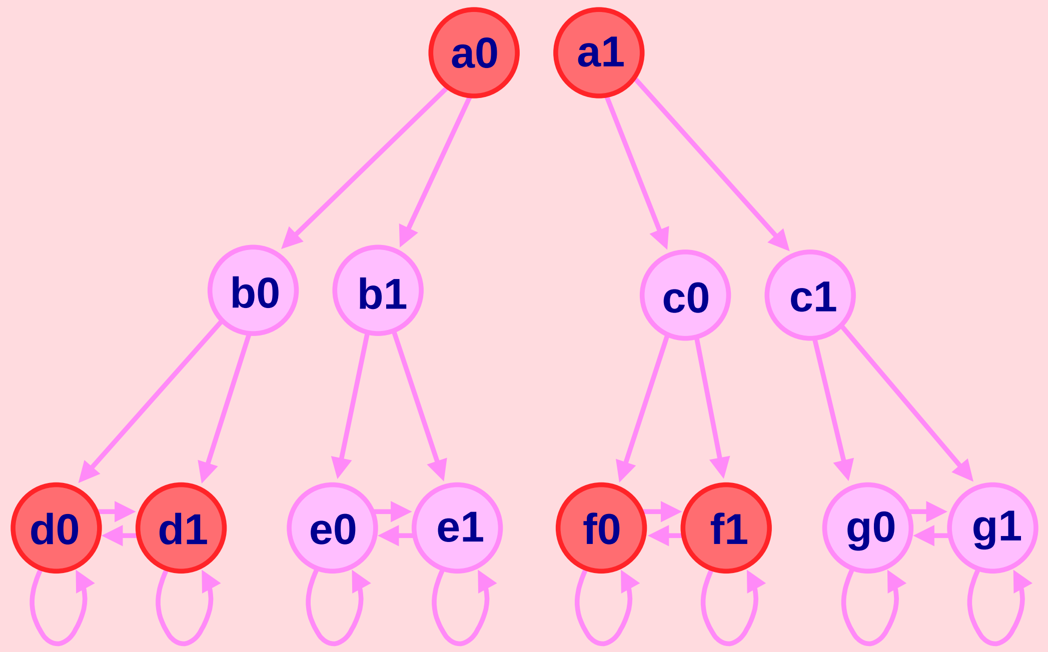
● states verifying $\exists_i EX\, p$



Possible CTL ambiguities

Checking the property $\forall_i AX \exists_i EX\, p$

🔴 states verifying $\forall_i AX \exists_i EX\, p$



Possible CTL ambiguities

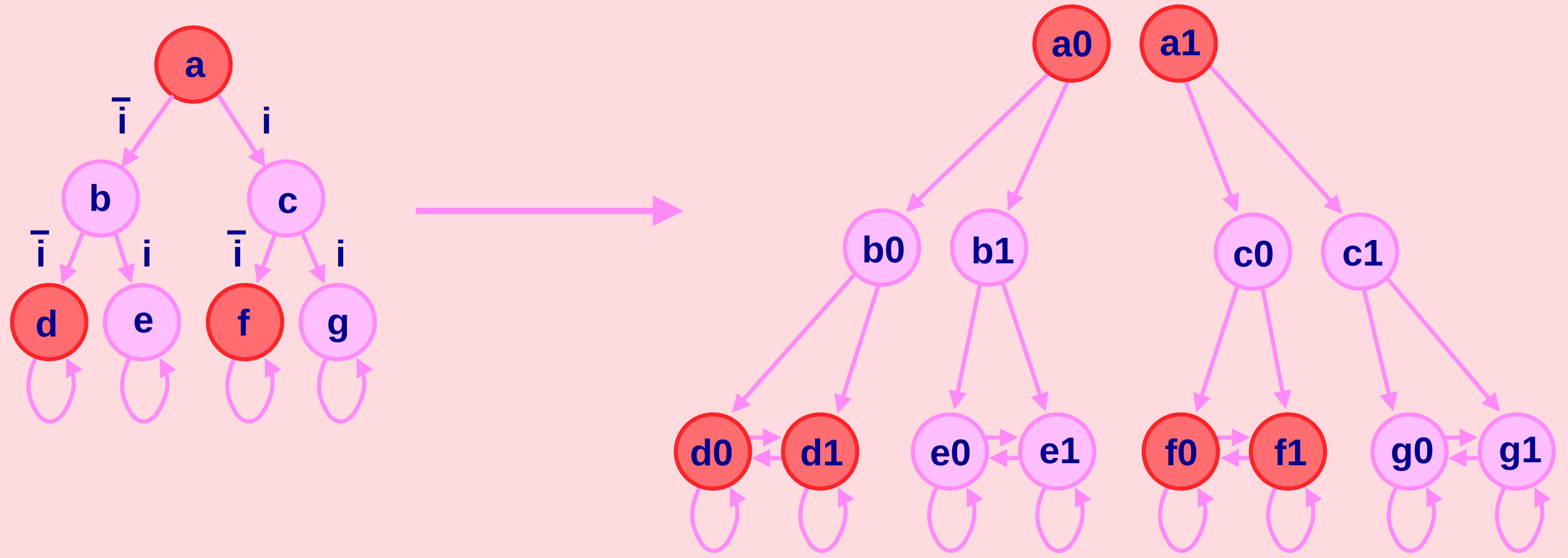
Comparison with AX EX p

Possible CTL ambiguities

Comparison with AX EX p

🟥 states verifying AX EX p

🟥 states verifying $\forall_i$ AX $\exists_i$ EX p

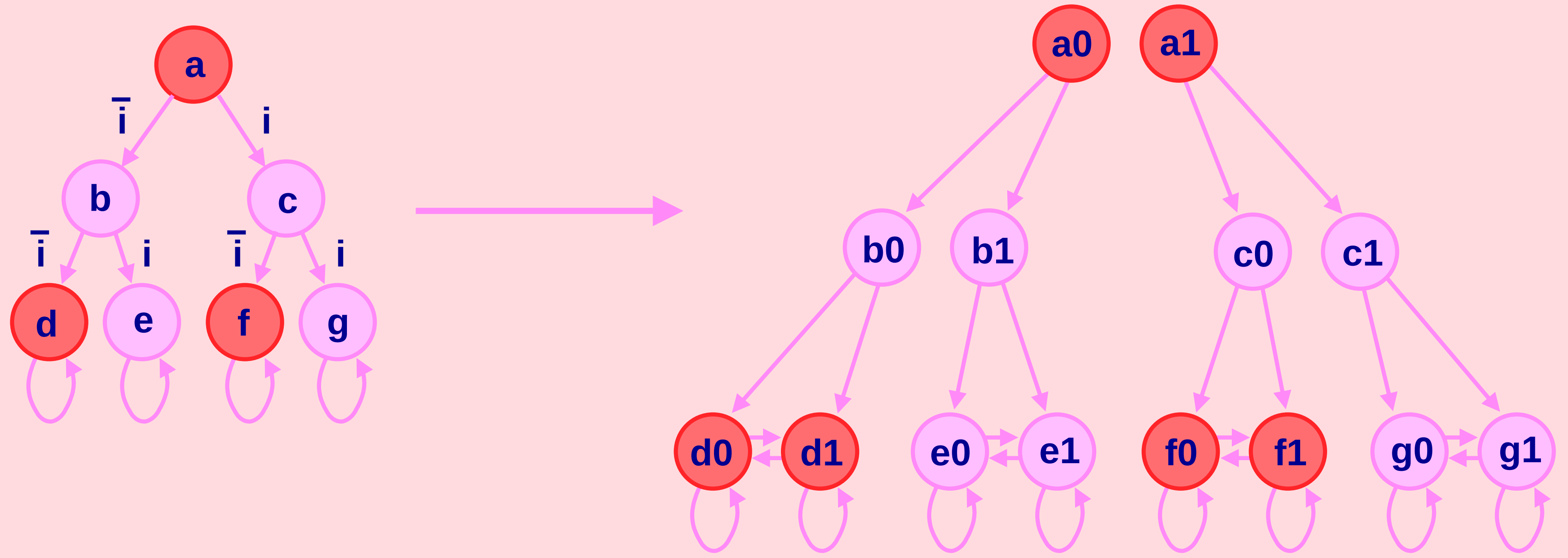


Possible CTL ambiguities

Comparison with AX EX p

🟥 states verifying AX EX p

🟥 states verifying $\forall_i$ AX $\exists_i$ EX p



The ambiguities have been removed

iCTL

iCTL

Extends CTL with $\forall_i$ and $\exists_i$

iCTL

Extends CTL with $\forall_i$ and $\exists_i$

More expressive than CTL

iCTL

Extends CTL with $\forall_i$ and $\exists_i$

More expressive than CTL

Easily integrable in a symbolic model−checker
(univ_abstract, exist_abstract)

iCTL

Extends CTL with $\forall_i$ and $\exists_i$

More expressive than CTL

Easily integrable in a symbolic model−checker
(univ_abstract, exist_abstract)

Applicable to Mealy machines

iCTL

Extends CTL with $\forall_i$ and $\exists_i$

More expressive than CTL

Easily integrable in a symbolic model−checker
(univ_abstract, exist_abstract)

Applicable to Mealy machines

$\forall_i$ and $\exists_i$ are not relevant for LTL

iCTL

Extends CTL with $\forall_i$ and $\exists_i$

More expressive than CTL

Easily integrable in a symbolic model−checker
(univ_abstract, exist_abstract)

Applicable to Mealy machines

$\forall_i$ and $\exists_i$ are not relevant for LTL

$\forall_i$ AX and $\exists_i$ EX seem similar to [*] and <*> of the mu−calculus

iCTL

Extends CTL with $\forall_i$ and $\exists_i$

More expressive than CTL

Easily integrable in a symbolic model−checker
(univ_abstract, exist_abstract)

Applicable to Mealy machines

$\forall_i$ and $\exists_i$ are not relevant for LTL

$\forall_i$ AX and $\exists_i$ EX seem similar to [*] and <*> of the mu−calculus

but what about $\forall_i$ EX ?

Thank you