Introduction to Formal Methods

Chapter 11. Abstraction Methods

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr

11. Abstraction Methods

- Abstraction Methods
  - A family of techniques used to simplify automata
  - Simplification aiming at verifying a system (faster) using a model checking approach

  - Examples:
    - (Pb1) " Does $A \models \phi$? " ← a complex problem
    - (Pb2) " Does $A' \models \phi'$? " ← a much simpler problem

  - " tricks of the trade "

- Organization of Chapter 11
  - When Is Model Abstraction Required?
  - Abstraction by State Merging
  - What Can Be Proved in the Abstract Automaton?
  - Abstraction on the Variables
  - Abstraction by Restriction
  - Observer Automata

11.1 When Is Model Abstraction Required?

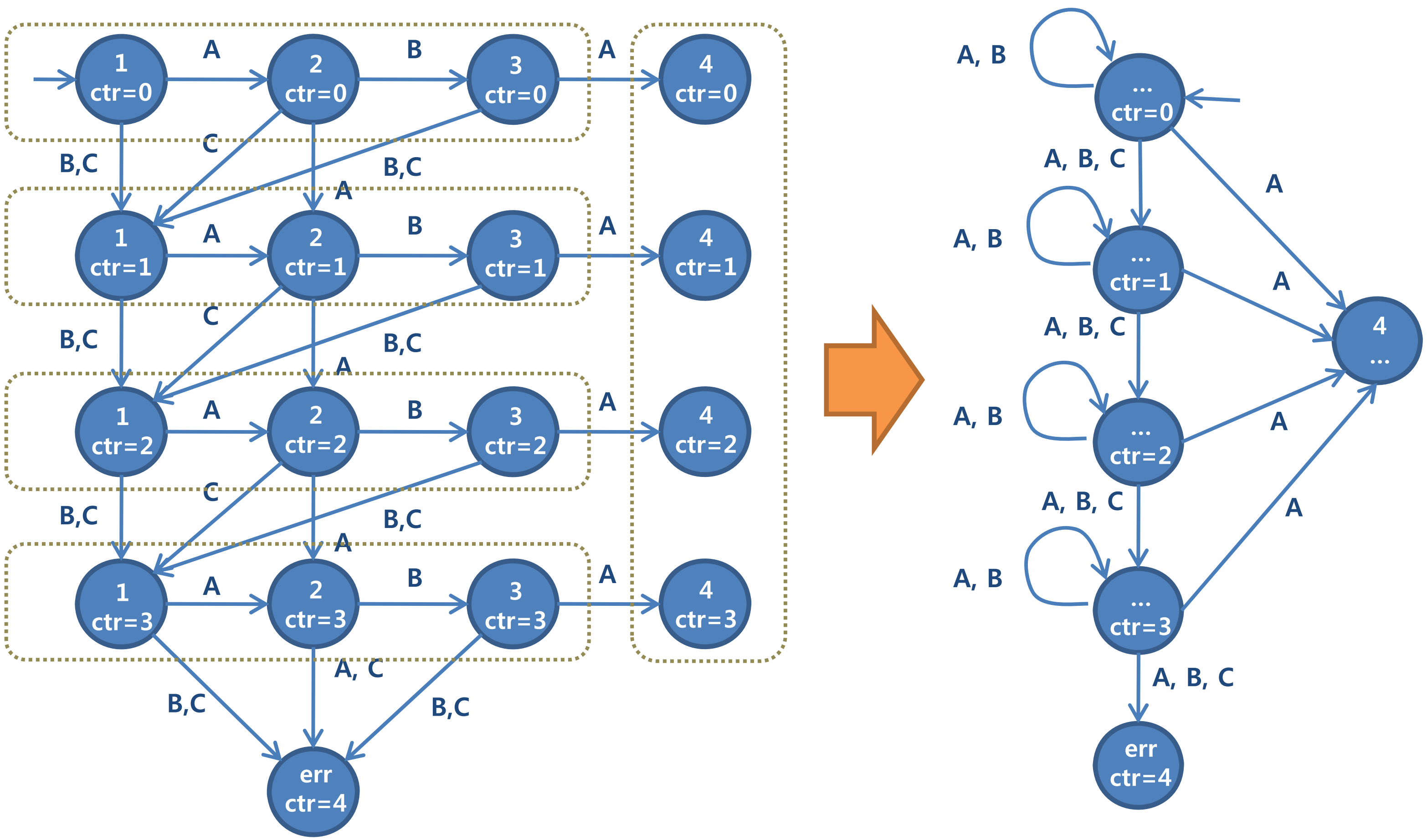
- Two main types of situations for model abstraction
  1. Size of the automaton
     - Too large :
     - Too many variables
     - Too many automata in parallel
     - Too many clocks in the timed automata
  2. Type of the automaton
     - Other types of automata
     - Using integer variables, communication channels, clocks, priorities, etc.

- Three classical abstraction methods
  1. Abstraction by State Merging
  2. Abstraction on the Variables
  3. Abstraction by Restriction

11.2 Abstraction by State Merging

- Folding
  - Viewing some states of an automaton as identical
  - The most important question : Correctness!
  - For example,
    - The digicode door lock with error counters (in Chapter 1)
    - Focusing on the error counter.
    - Correctness problem:
      - All states in A' can be reached through the letter A, but not in A

A


1 ctr=0
2 ctr=0
3 ctr=0
4 ctr=0
1 ctr=1
2 ctr=1
3 ctr=1
4 ctr=1
1 ctr=2
2 ctr=2
3 ctr=2
4 ctr=2
1 ctr=3
2 ctr=3
3 ctr=3
4 ctr=3
err ctr=4
A
B
C
B,C
A, C
… ctr=0
… ctr=1
… ctr=2
… ctr=3
4 …
err ctr=4
A, B
A, B, C

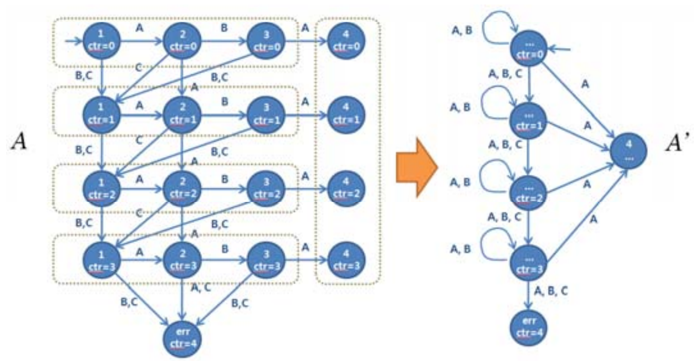
A'

11.3 What Can be Proved in the Abstract Automaton?

- We can use <u>state merging</u> to verify <u>safety properties</u>

- Observation (Merging states from A to A')
  1. A' has more behaviors than A.
  2. Now the more behaviors an automaton has, the fewer safety properties it fulfills.
  3. Thus, if A' satisfies a safety property ϕ then a fortiori A satisfies ϕ.
  4. However, if A' does not satisfy ϕ, no conclusions can be drawn about A.

- More behaviors
  - A' has more behaviors than A
  - All executions of A remain present (in folded form) in A'
  - Some new behaviors may be introduced in A'
    - For example, many infinite loops are possible in A'



- Preserving safety properties
  - Necessary to ensure that the property ϕ is indeed a safety property.

- One-way preservation
  - If A' does not satisfies ϕ , then A' satisfies $\neg\phi$.
  - But, in general the negation of a safety property is not a safety property.
  - Abstraction methods are often one-way:
    - If the answer is positive, then is positive too.
    - If the answer is negative, then we learned nothing about A.

- Some necessary precautions
  - Skipped.
  - about the propositions' merging and marking in model checking algorithms

- Modularity
  - State merging is preserved by product.
  - $A' \parallel B$ can be obtained from $A \parallel B$ by a merging operation

- State merging in practice
  - Question : " How will we guess and then specify the sets of states to be merged ? "
  - Answer : " The user is the one who defines and applies his own abstraction. "
    " No tool assistance is offered. "

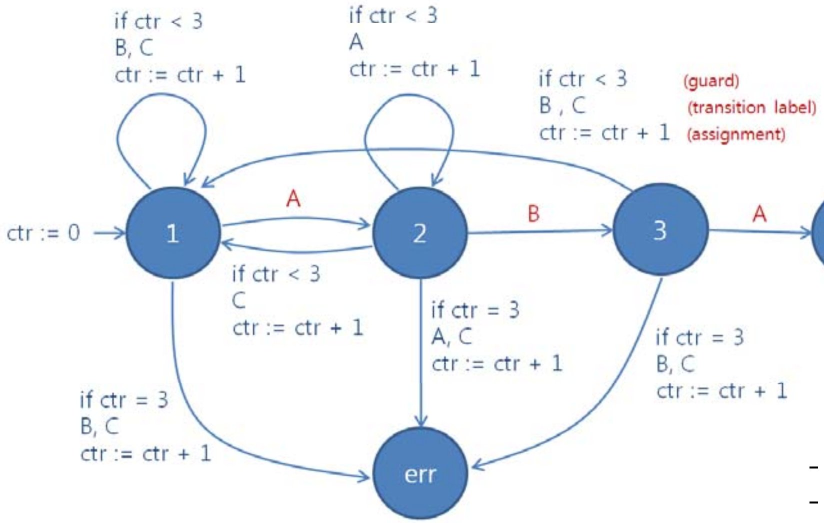
  → Abstraction on variables are often easy to define and implement.

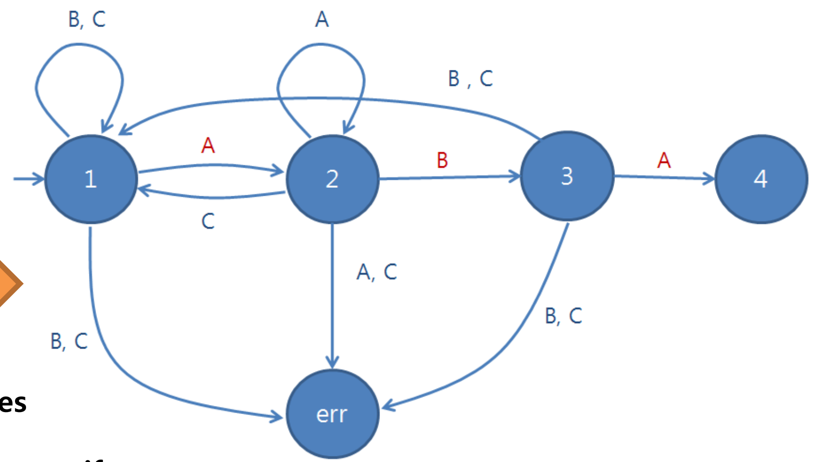
11.4 Abstraction on the Variables

- Abstraction on the variables
  - Concerns the “data” part of automata with variables
  - Directly applies to the description of the automata with variables

- Example

var ctr: int;

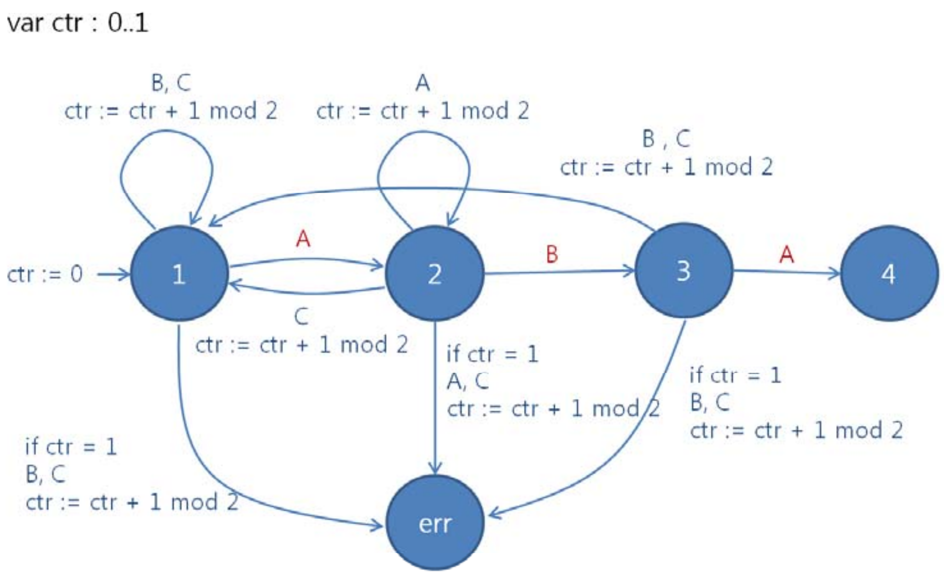




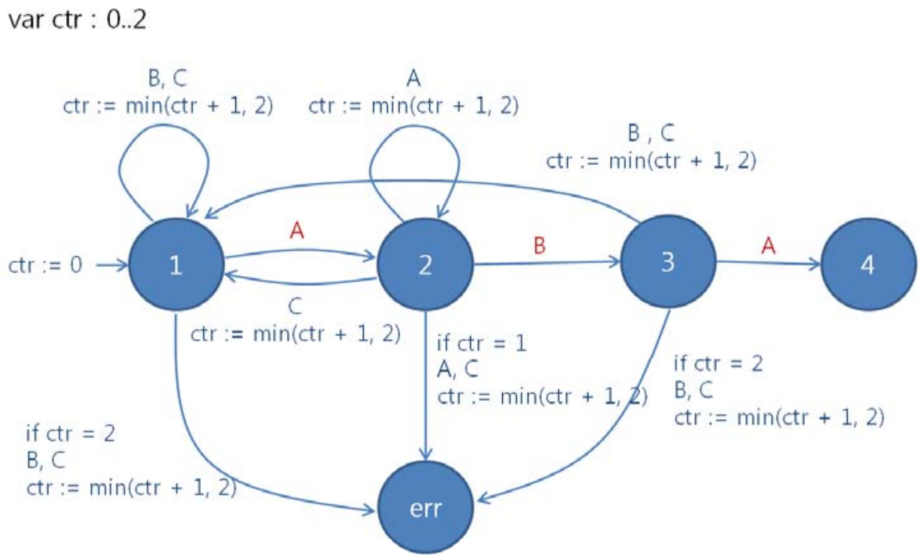
- **Deleting variables**
- **Simple**
- **But, too coarse to verify**

- Abstraction differs from deletion
  - Abstract Interpretation
    - Mathematical theory aiming at defining, analyzing, justifying methods based on abstration

- Bounded variables
  - Narrow down the domain of variables
  - For example,
    - Integer → 0 ~ 10 value
    - The digicode with a modulo 2 counter



The digicode with a modulo 2 counter

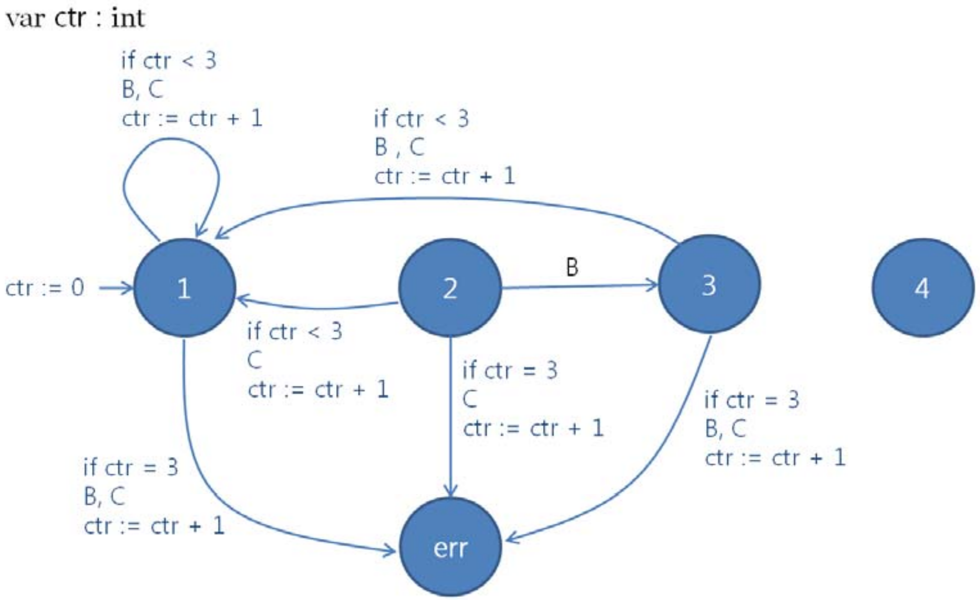


The digicode with a counter bounded by 2

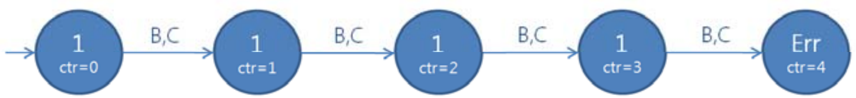
13.5 Abstraction by Restriction

- Restriction
  - A particular form of simplification
  - Operates by forbidding some behaviors of the system or by making some impossible
    - Removing states or transitions
    - Strengthening the guard, etc.

  - For example
    - Remove all the transitions labeled A



The digicode with no A transition

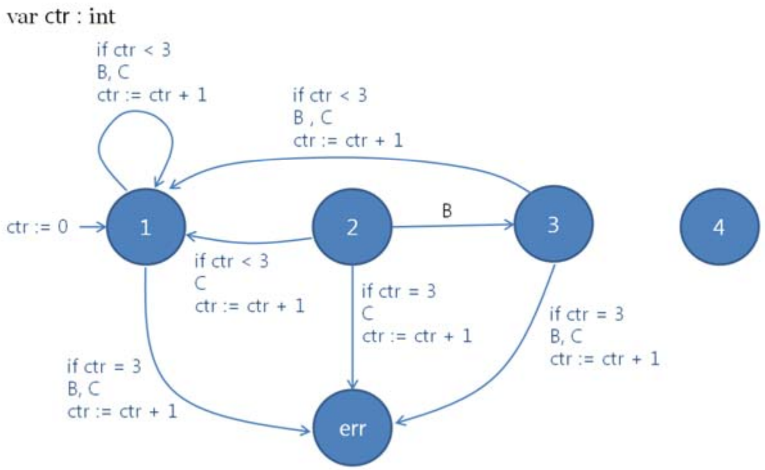


The unfolding of the digicode with no A transition

- What the restrictions preserve

  - If A' is obtained from A by restriction, then literally all the behaviors of A' are behaviors of A.
  - Thus if A' does not satisfy a safety property, then a fortiori neither does A.
  - Conditional reachability property " EF err " = negation of safety property

  - For example,
    - A' satisfies EF err
    - So we conclude that A also satisfies this property

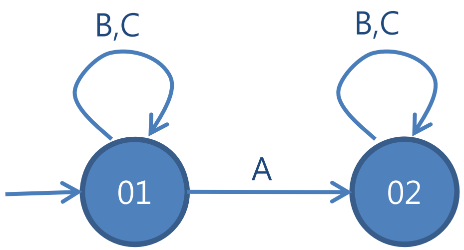


  - Inverse preservation
    - A safety property does not hold.
      (To find errors)
    - But, not to prove the correctness of A

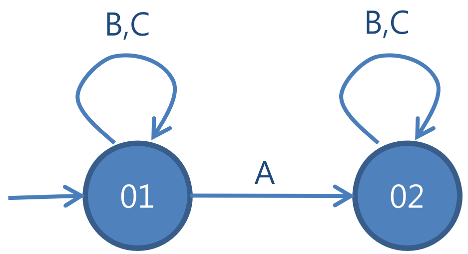
  - Advantage of restriction
    - Simplicity in conceptual and implementational
    - It is a modular operation
    - It naturally applies to an automaton with variables

11.6 Observer Automata

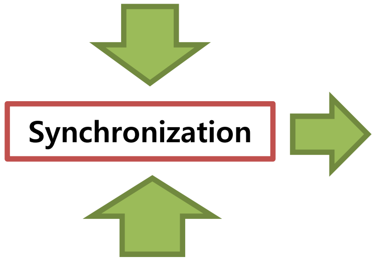
- Observer automata
  - Aiming at simplifying a system by restricting its legitimate <u>behaviors</u> to those accepted by an automata outside the system, called observer automata.
  - Reduce the size of automata by restricting its behavior rather than its structure (states and transitions in restriction methods)
  - PLTL model checking algorithm (in Chapter 3) use the concept.
  - An example
    - Supposed that a single A may occur to prove the property.



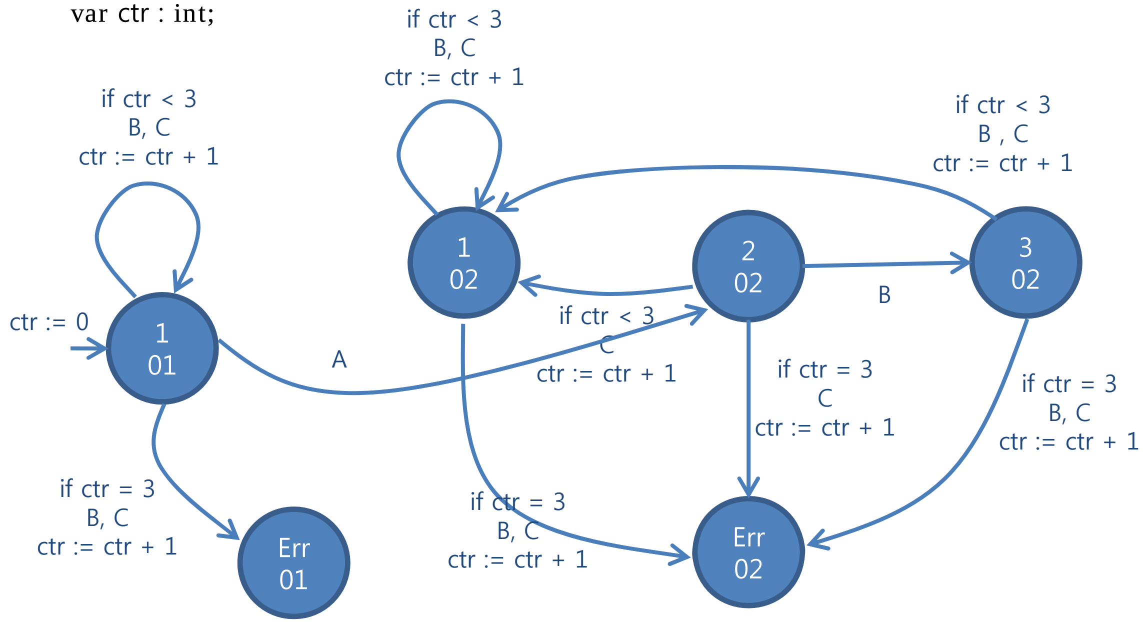
An observer automaton O



An observer automaton O

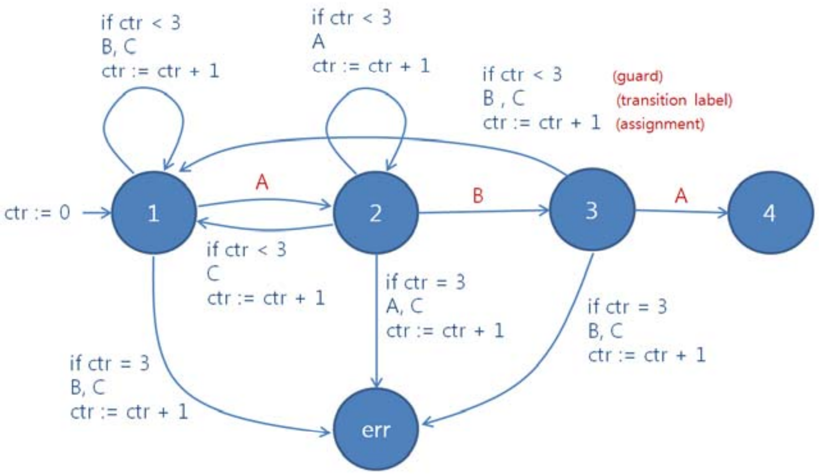


var ctr : int;



The synchronized digicode with its observer

var ctr: int;



An automaton A for the digicode