Chapter 2
Modeling System

Formal Model

- Construct a formal model of a system
  - Specifying key properties of the system
  - Abstract away details
  - For digital circuits
    - Useful: gates and Boolean values
    - Useless: voltage levels
  - For communication protocol
    - Useful: exchange of messages
    - Useless: contents of messages

- Modeling reactive system and their behavior over time
  - Interact with their environment frequently and often do not terminate
- State captures the values of the variables at a particular instant of time
- A transition describe the change by giving the state before the action occurs and the state after the action occurs.
- A computation is an infinite sequence of state where each state is obtained from the previous state by some transition

- State transition system: A **Kripke** structure
  - Capture the behavior of reactive system.
- Path: modeling computations of a system.
- Concurrent system
  - Program
  - Diagram for a circuit
- Unifying formalism to represent a current system
  - First order logic
  - Extract the Kripke structure

- Definition of Kripke structure
- How to extract such structure from first order logic
- How difference programming constructs can be represented in term of first order formulae

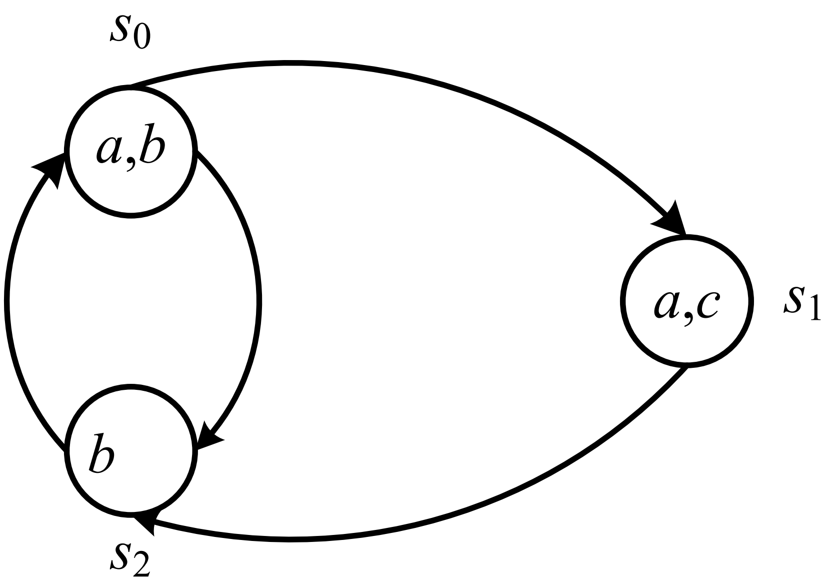
Kripke structure(KS)

- Let AP be a set of atomic propositions. A Kripke structure M over AP is a four tuple M=(S,S_0,R,L) where
  - S is a finite set of states
  - $S_0 \subseteq S$ is the set of initial states
  - R $\subseteq S \times S$ is a transition relation that must be total
    - Total: Each state s∈S, existing a state s′∈S, such that R(s, s′)
  - L: $S \rightarrow 2^{AP}$ is a function that labels each state with the set of atomic propositions true in that state
- Sometimes ignore the initial states S_0

Modeling concurrent systems (cont)

- A path in the structure M from a state s is an infinite sequence of states $\pi = s_0 s_1 s_2 \ldots$ such that $s_0 = s$ and $R(s_i, s_{i+1})$ holds for all $i \geq 0$

example



- KS structure M=(S,S_0,R,L) over AP, AP={a,b,c}
  - S={s_0, s_1,s_2},

  ……

First order logic

- First order logic: logical connectives and quantifications
- Describe states of concurrent system with first order logic
  - V={$v_1, \ldots, v_n$}is the set of system variables.
  - Variables in V range over a finite set D
  - A valuation for V is a function s: V→D
  - A state is described by giving values for all of the elements in V
  - State: Write a formula that is true for that valuation

First order logic(cont)

- Describe states of concurrent system with first order logic(cont)
  - Example V={v_1,v_2,v_3} and a valuation<$v_1 \leftarrow 2$, $v_2 \leftarrow 3$, $v_3 \leftarrow 5$>
  - Derive the formula $(v_1=2) \wedge (v_2=3) \wedge (v_3=5)$
  - A formula may be true for many valuations, then representing many states
  - Initial: a first order formula S_0

First order logic(cont)

- transition of concurrent system: first order logic
  - A set of ordered pairs of states
  - Two system variables V and V′.
  - Variables in V are present states
  - Variables in V′ are next states
  - Each variable *v* in V has a corresponding next state variable in V′, denoted by *v*′
  - Transition: an ordered pair of states of valuations in V and V′
    - represent the transition by formulas
    - a transition relation: the set of ordered pairs of states
    - R(V, V′) denotes a formula that represents transition relation

First order logic(cont)

- Example of transition system
  - Let V={v_1,v_2,v_3} and a valuation<$v_1 \leftarrow 2$, $v_2 \leftarrow 3$, $v_3 \leftarrow 5$>
  - V'={v_1',v_2',v_3'} and a valuation < $v_1' \leftarrow 1$, $v_2' \leftarrow 5$, $v_3' \leftarrow 4$ >
  - A transition:
    - $(v_1=2 \wedge v_2=3 \wedge v_3=5) \wedge (v_1'=1 \wedge v_2'=5 \wedge v_3'=4)$

First order logic(cont)

- Describe atomic propositions AP
  - AP: v=d where $v \in$V and d $\in$D
  - A proposition v=d will be true in a state s if $s(v)$=d

Construct a KS from the first order formula

- Initial state S_0 and transition R
  - The set of states S is the set of all valuations for V.
  - The set initial S_0 is the set of all valuations s_0 for v that satisfy the formula S_0
  - Let s and s' be two states, then R(s,s') holds if R evaluates to True when each $v \in V$ is assigned the value $s(v)$ and each $v' \in V'$ is assigned the value $s'(v')$
  - The labeling function $L: S \rightarrow 2^{AP}$ is defined so that L(s) is the subset of all atomic propositions true in s.
  - Add R(s,s)if some state s has no successor so that a KS is total.

Construct a KS

- Example
  - $V=\{x,y\}$ and $D=\{0,1\}$
  - $S_0(x,y) \equiv x=1 \wedge y=1$
  - Transition:

$R(x,y,x',y') \equiv x'=(x+y) \bmod 2 \wedge y'=y$

Question: $KS=(S,S_0,R,L)$??

Concurrent System

- Consider two modes of execution
  - Digital circuit
    - Asynchronous or interleaved execution: only one component makes a step at a time
    - Synchronous: all of the components make a step at the same time.
  - Program
    - Communication by shared variables

Digital circuits

- Each state holding element of a circuits can have the value 0 or 1
- Give a valuation, we can write a boolean expression that is true .
- $S_0(V)$ and $R(V, V')$ represent the set of initial states and the transition relation of the circuit.

Synchronous digital circuits

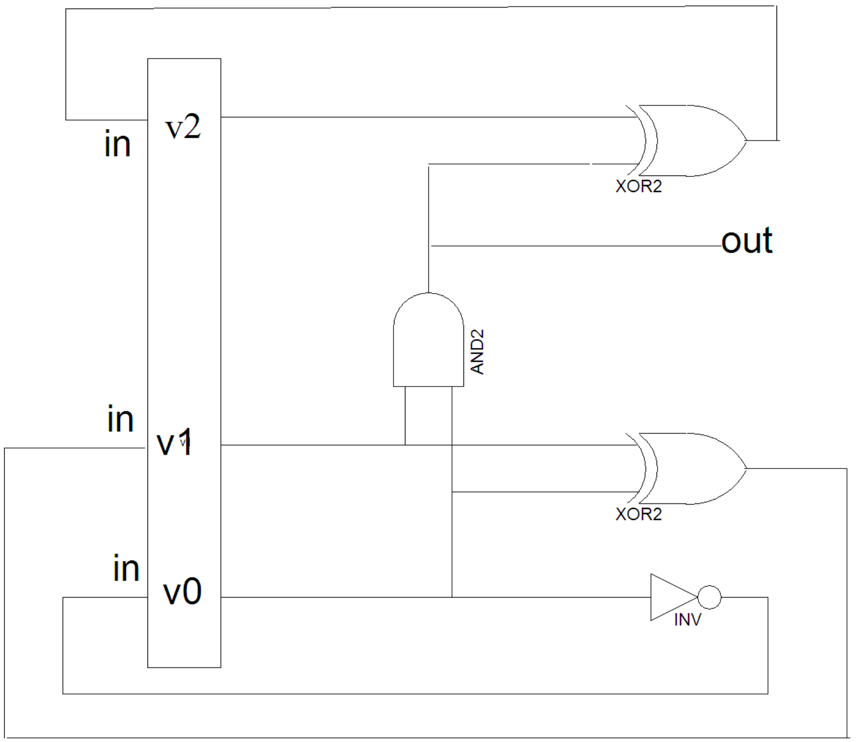
- Let V=$\{v_0, v_1, \ldots, v_{n-1}\}$ and

  V′ =$\{v_0', v_1', \ldots, v_{n-1}'\}$
- $v_i' = f_i(\mathrm{V})$
- Define the relations
  - $\mathrm{R}_{\mathrm{i}}(\mathrm{V}, \mathrm{V}') \equiv (v_i' \Leftrightarrow f_i(\mathrm{V}))$
  - Conjunction of these formulae

$$\mathrm{R}(\mathrm{V}, \mathrm{V}') \equiv \mathrm{R}_0(\mathrm{V}, \mathrm{V}') \wedge \mathrm{R}_0(\mathrm{V}, \mathrm{V}') \wedge \ldots \wedge \mathrm{R}_{\mathrm{n}-1}(\mathrm{V}, \mathrm{V}')$$

- Question: synchronous model of 8 counter?

Synchronous digital circuits



- Synchronous modulo 8 counter
- Three components, they compute their valuation simultaneously every time.

Asynchronous digital circuit

- Assume that all the components of circuits have exactly one output and have no internal state variables
- Use an interleaving semantics in which exactly one component changes at a time
- The results in a disjunction of the form

$$R(V, V') \equiv R_0(V, V') \vee R_1(V, V') \vee \ldots \vee R_{n-1}(V, V')$$

where

$$R_i(V, V') \equiv (v_i' \Leftrightarrow f_i(V)) \wedge \bigwedge_{j \neq i} (v_j' \Leftrightarrow v_j)$$

Difference between synchronous and asynchronous

- $V=\{v_0,v_1\}$
- $v_0{}'=v_0 \oplus v_1$ and $v_1{}'=v_0 \oplus v_1$
- For synchronous, the transition relation is

$$R(V,V') \equiv (v_0{}' \Leftrightarrow v_0 \oplus v_1) \wedge (v_1{}' \Leftrightarrow v_0 \oplus v_1)$$

- For asynchronous, the transition relation is

$$R(V,V') \equiv ((v_0{}' \Leftrightarrow v_0 \oplus v_1) \wedge (v_1{}' \Leftrightarrow v_1)) \vee \\ ((v_0{}' \Leftrightarrow v_0) \wedge (v_1{}' \Leftrightarrow v_0 \oplus v_1))$$

Difference between synchronous and asynchronous

- If a state(v_0=1,v_1=1)
  - For synchronous, the next state is (v_0=0,v_1=0)
  - For asynchronous, the next states have two states:(v_0=0,v_1=1)and (v_0=1,v_1=0)
- Question: Models of synchronous and asynchronous?

Program modeling

- Sequential program
  - Each statement has a unique entry point and unique exit point that are labeled.
  - A labeling transformation that given an unlabeled program P results in a labeled program P^L
  - A statement exit point is the entry point of the next statement

Program modeling (cont)

- Sequential program (cont)
  - ◆ Define the labeled statement P^L:
    - If P is not a composite statement, then $P^L = P$
    - If P=P1;P2, then $P^L = P_1^L;\ l'':\ P_2^L$.
    - If P=if b then P_1 else P_2 endif, therefore

      P^L=if b then l_1: P_1^L else l_2:P_2^L endif.
    - If P=while b do P_1 endwhile, then

  P^L= while b do l_1:P_1^L endwhile

Program modeling (cont)

- **Sequential program (cont)**
  - pc—program counter that range over the set of program labels and additional value ⊥(undefined value, the program is not active)
  - The entry and exit point of P are labeled by *m* and *m′*
  - *same(Y)* is an abbreviation for the formula

$$\bigwedge_{y\in Y}(y'=y)$$

Program modeling (cont)

- Sequential program (cont)
  - Initial states:

    pre(V): initial value of the variables of P

    Initial states: $S_0(V, pc) \equiv pre(V) \wedge pc=m$
  - Transition procedure $C\,(l, P, l')$: the entry label l, the labeled statement P and the exit label l'
  - $C\,(l, P, l')$ is the disjunction of all transition

Program modeling (cont)

- Sequential program (cont)
  - ◆ *C* define statements′ transition
    - ▪ assignment:

$C\,(l, v \leftarrow \mathrm{e}, l') \equiv \mathrm{pc}=l \wedge \mathrm{pc}'=l' \wedge v'=\mathrm{e} \wedge \mathit{same}(\mathrm{V} \backslash \{v\})$

- ▪ skip:

$C\,(l, \mathrm{skip}, l') \equiv \mathrm{pc}=l \wedge \mathrm{pc}'=l' \wedge \mathit{same}(\mathrm{V})$

- ▪ Sequential composition :

$C\,(l, \mathrm{P}_1; l'': \mathrm{P}_2, l') \equiv C\,(l, \mathrm{P}_1, l'') \vee C\,(l'', \mathrm{P}_2, l')$

- ▪ conditional:
- ▪ while:

Program modeling (cont)

- Sequential program (cont)
  - C define statements' transition
    - conditional:

      $C\,(l, \text{if b then } l_1 : \mathrm{P}_1 \text{ else } l_2 : \mathrm{P}_2 \text{ endif}, l')$

      is the disjunction of the following four formulae:

      - ✓ $\mathrm{pc}=l \wedge \mathrm{pc}'=l_1 \wedge \mathrm{b} \wedge \mathit{same}(\mathrm{V})$
      - ✓ $\mathrm{pc}=l \wedge \mathrm{pc}'=l_2 \wedge \neg\mathrm{b} \wedge \mathit{same}(\mathrm{V})$
      - ✓ $C\,(l_1, \mathrm{P}_1, l')$
      - ✓ $C\,(l_2, \mathrm{P}_2, l')$
    - while:

Program modeling (cont)

- Sequential program (cont)
  - C define statements' transition
    - while: C (l, while b do l_1: P1 endwhile, l')

      is the disjunction of the following three formulas:
      - ✓ $\mathrm{pc}=l \wedge \mathrm{pc}'=l_1 \wedge \mathrm{b} \wedge \mathit{same}(\mathrm{V})$
      - ✓ $\mathrm{pc}=l \wedge \mathrm{pc}'=l' \wedge \neg \mathrm{b} \wedge \mathit{same}(\mathrm{V})$
      - ✓ $C\,(l_1, \mathrm{P}_1, l\,)$

exercise

- building boolean formula for the following program, while

V={x,y,z}, initial value: x=y=z=0

- Program

```
 x=y+1;z=z+2;
For(y;y<=3;y++)
  if x<y then x++;else y++;
```

experiment

IMP language:

- Aexp

  $a ::= n \mid x \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$, $n \in [0,2]$

- Bexp

  $b ::= \mathrm{true} \mid \mathrm{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b$

  $\mid b_0 \wedge b_1 \mid b_0 \vee b_1$

- Com

  $c ::= \mathrm{skip} \mid x := a \mid c_0 ; c_1 \mid \mathrm{if}\ b\ \mathrm{then}\ c_0\ \mathrm{else}\ c_1$

  $\mid \mathrm{while}\ b\ \mathrm{do}\ c$

Experiment (cont’)

  - The range of variables integers of [0,2]
  - Values of boolean variables: 0 and 1
  - The name of all variables are single lower letter
- Give a program of IMP, Please transform it into labeled program
- Please translate a program of IMP into the first order formulae

Concurrent programs

- A set of processes that can be executed in parallel.
- A process is composed of sequential statements
- asynchronous programs in which exactly one process can be executed at any time
- V_i is the set of variables in the process P_i, pc_i is the program counter of P_i, PC is the set of all program counter

Concurrent programs(cont)

- A concurrent program P has the form:

Cobegin P_1|| P_2 || … || P_n coend

- The labeled concurrent program P^L:
  - If P= Cobegin P_1|| P_2 || … || P_n coend, then

P^L= conbegin $l_1 : P_1^L\ l_1' \| l_2 : P_2^L\ l_2' \| \ldots \| l_n : P_n^L\ l_n'$ coend

Concurrent programs(cont)

- Initial and transition formulae
  - Initial state set

$$S_0(V,PC)\equiv \mathrm{pre}(V) \wedge \mathrm{pc} = m \wedge \bigwedge_{i=1}^{n} (\mathrm{pc}_i=\bot)$$

  - Transition procedure

$$C\,(m, \mathrm{conbegin}\ l_1{:}\mathrm{P}_1^{L}\ l_1' \,\|\, l_2{:}\mathrm{P}_2^{L}\ l_2' \,\|\ \ldots\ \|\, l_n{:}\mathrm{P}_n^{L}\ l_n'\,\mathrm{coend}, m'\,)$$

is the disjunction of three formulas:

- $\mathrm{pc}=m \wedge \mathrm{pc}_1'=l_1 \wedge \ldots \wedge \mathrm{pc}_n'=l_n \wedge \mathrm{pc}' =\bot$
- $\mathrm{pc}=\bot \wedge \mathrm{pc}_1=l_1\,' \wedge \ldots \wedge \mathrm{pc}_n=l_n\,' \wedge \mathrm{pc}' =m\,' \wedge \bigwedge_{i=1}^{n} (\mathrm{pc}_i'= \bot)$
- $\vee_{i=1}^{n} (\ C\,(l_i\,, \mathrm{P}_i, l_i') \wedge \mathit{same}(V\backslash V_i) \ \wedge \mathit{same}(PC \backslash\{\mathrm{pc}_i\}))$

Concurrent programs(cont)

- Shared variables
  - wait: $C\,(l, \text{wait}(b), l')$ is a disjunction of the following two formulae:
    - $(pc_i=l \wedge pc_i'=l \wedge \neg b \wedge \mathit{same}(V_i))$
    - $(pc_i=l \wedge pc_i'=l' \wedge b \wedge \mathit{same}(V_i))$
  - lock: $C\,(l, \text{lock}(v), l')$ is a disjunction of the following two formulae:
    - $(pc_i=l \wedge pc_i'=l \wedge v=1 \wedge \mathit{same}(V_i))$
    - $(pc_i=l \wedge pc_i'=l' \wedge v=0 \wedge v'=1 \wedge \mathit{same}(V_i \setminus \{v\}))$
  - unlock:
    - $C\,(l, \text{unlock}(v), l')$

      $\equiv pc_i=l \wedge pc_i'=l' \wedge v'=0 \wedge \mathit{same}(V_i \setminus \{v\})$

Concurrent programs(cont)

- Example
  - P≡m: cobegin P_0 || P_1 coend m′
  - two processes P_0 and P_1.

```
$P_0$::
 $l_0$:    while True do
            $NC_0$: wait(turn=0);
           $CR_0$: turn=1;
            endwhile;
$l_0'$
```

```
$P_1$::
 $l_1$:    while True do
            $NC_1$: wait(turn=1);
           $CR_1$: turn=0;
            endwhile;
$l_1'$
```

Notation: For P_i, CR_i: turn=1 ≡ CR_i: turn=(turn+1) mod 2

Concurrent programs(cont)

- **Initial states of P**

$S_0(V, PC) \equiv pc=m \wedge pc_0=\perp \wedge pc_1=\perp$

- **Transition relation R(V, PC, V′, PC′)**
  - $pc=m \;\wedge\; pc_0'=l_0 \wedge pc_1'=l_1 \;\wedge pc'=\perp$
  - $pc=\perp \wedge pc_0=l_0' \wedge pc_1=l_1' \wedge pc' =m' \wedge\; pc_0'=\perp \wedge pc_1'=\perp$
  - $C(l_0,P_0,l_0') \wedge same(V\backslash V_0) \wedge same(PC\backslash\{pc_0\})$,

  which is equivalent to

  $C(l_0,p_0,l_0') \wedge same(pc,pc_1)$

  - $C(l_1,P_1,l_1') \wedge same(V\backslash V_1) \wedge same(PC\backslash\{pc_1\})$,

  which is equivalent to

  $C(l_1,p_1,l_1') \wedge same(pc,pc_0)$

Concurrent rograms(cont)

- Summary: C(m, cobegin $P_0 \| P_1$ coend, m′)
  - $pc=m \wedge pc_0'=l_0 \wedge pc_1'=l_1 \wedge pc'=\perp$
  - $pc=\perp \wedge pc_0=l_0' \wedge pc_1=l_1' \wedge pc'=m' \wedge pc_0'=\perp \wedge pc_1'=\perp$
  - $C(l_0,p_0,l_0') \wedge same(pc,pc_1)$
  - $C(l_1,p_1,l_1') \wedge same(pc,pc_0)$

Concurrent programs(cont)

- Transition relation $R(V, PC, V', PC')$(cont) . For each processes P_i, $C(l_i, p_i, l_i')$ is the disjunction of
- $C(l_i$, while True do NC_i…endwhile, $l_i')$
    - $pc_i=l_i \wedge pc_i'=NC_i \wedge \text{True} \wedge \text{same(turn)}$
    - $pc_i=l_i \wedge pc_i'=l_i' \wedge \text{False} \wedge \text{same(turn)}$
    - $C(NC_i$, ,wait(turn=i);CR_i:turn=(i+1)mod 2, $l_i)$

Concurrent programs(cont)

- C(l_i, while True do …endwhile, l_i')
  - ……
  - C(NC_i, ,wait(turn=i);CR_i:turn=(i+1)mod 2, l_i)
  - C(NC_i, ,wait(turn=i),CR_i)$\vee$C(CR_i, turn=$(i+1)$mod 2, l_i)
  - C(NC_i, ,wait(turn=i),CR_i)
  - $\mathrm{pc}_\mathrm{i}=NC_i \wedge \mathrm{pc}_\mathrm{i}'=CR_i \wedge \mathrm{turn}=i \wedge$ same(turn)
  - $\mathrm{pc}_\mathrm{i}=NC_i \wedge \mathrm{pc}_\mathrm{i}'=NC_i \wedge \mathrm{turn}\neq i \wedge$same(turn)
  - C(CR_i, turn=$(i+1)$mod 2, l_i)
  - $\mathrm{pc}_\mathrm{i}=CR_i \wedge \mathrm{pc}_\mathrm{i}'=l_i \wedge \mathrm{turn}' =(i+1)$ mod 2

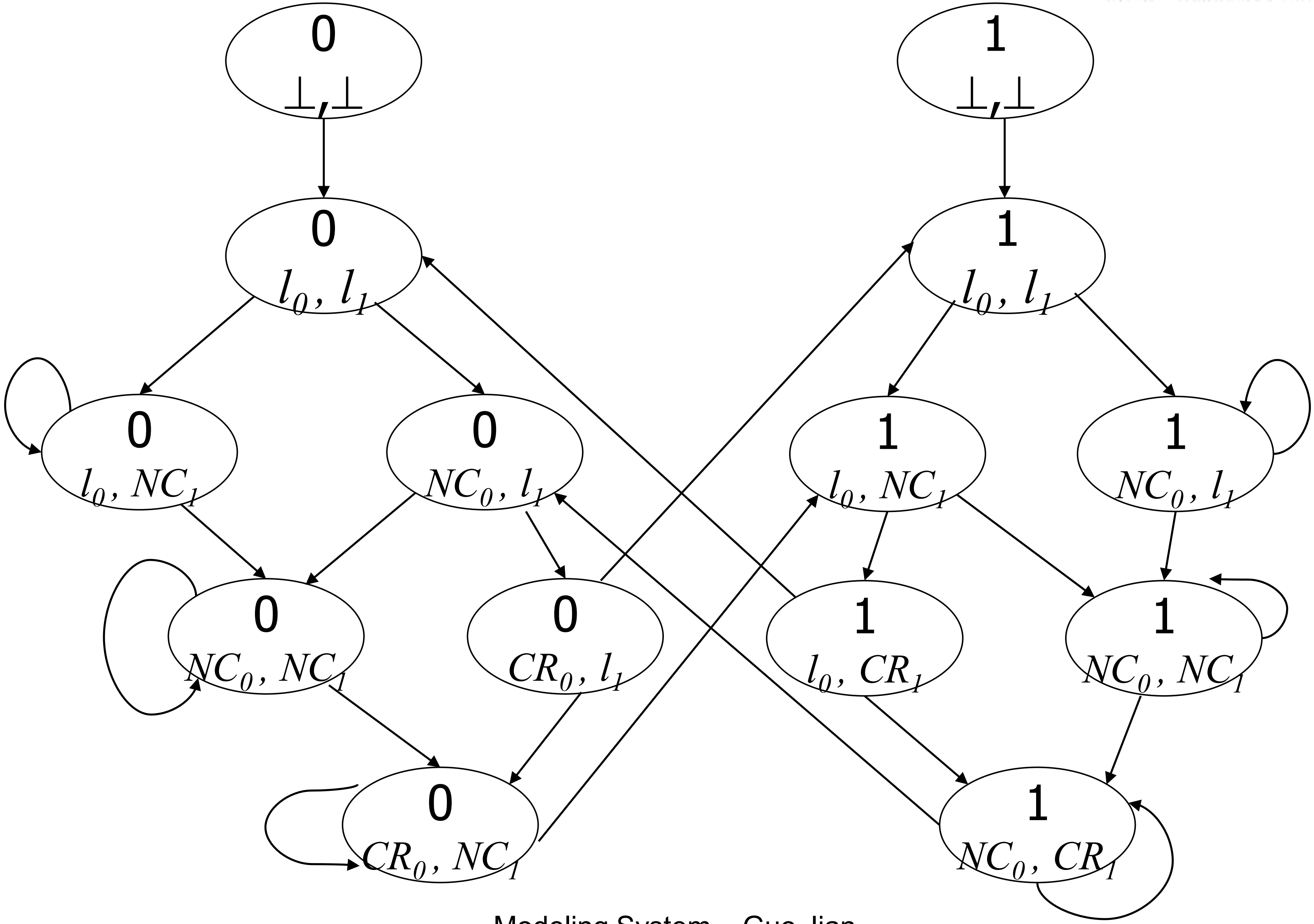
Concurrent programs(cont)

- C(l_i, while True do …endwhile, l_i')
  - $pc_i = l_i \wedge pc_i' = NC_i \wedge \text{True} \wedge \text{same(turn)}$
  - $pc_i = NC_i \wedge pc_i' = CR_i \wedge \text{turn} = i \wedge \text{same(turn)}$
  - $pc_i = NC_i \wedge pc_i' = NC_i \wedge \text{turn} \neq i \wedge \text{same(turn)}$
  - $pc_i = CR_i \wedge pc_i' = l_i \wedge \text{turn}' = (i+1) \bmod 2$

Concurrent programs(cont)

- Summary: C(m, cobegin $P_0 \| P_1$ coend, m′)
  - $pc=m \wedge pc0'=l_0 \wedge pc1'=l_1 \wedge pc'=\perp$
  - $pc=\perp \wedge pc_0=l_0' \wedge pc_1=l_1' \wedge pc'=m' \wedge pc_0'=\perp \wedge pc_1'=\perp$
  - $pc_i=l_i \wedge pc_i'=NC_i \wedge \mathrm{True} \wedge \mathrm{same(turn)}$
  - $pc_i=NC_i \wedge pc_i'=CR_i \wedge turn=i \wedge \mathrm{same(turn)}$
  - $pc_i=NC_i \wedge pc_i'=NC_i \wedge turn\neq i \wedge \mathrm{same(turn)}$
  - $pc_i=CR_i \wedge pc_i'=l_i \wedge turn'=(i+1) \bmod 2$

Kripke Structure



exercise

- $P \equiv m$: cobegin P_0 || P_1 coend m'
- two processes P_0 and P_1.

```
$P_0$::
 $l_0$:    while True do
       $NC_0$: wait(turn=0);
       $AS_0$:   x=1;
       $CR_0$: turn=1;
         endwhile;
$l_0'$
```

```
$P_1$::
 $l_1$:    while True do
       $NC_1$: wait(turn=1);
       $AS_1$:       x=2;
       $CR_1$: turn=0;
         endwhile;
$l_1'$
```

where: turn, x are integers and turn$\in$[0,1] and x $\in$[1,2]

The initialized value of x is 1.

1. First order logic formulas
2. Kripke Structure.

experiment

IMP language:

- Aexp $a ::= n | X | a_0 + a_1 | a_0 - a_1 | a_0 \times a_1$, $n \in [0,2]$
- Bexp

  $b ::= true | false | a_0 = a_1 | a_0 \leq a_1 | \neg b$

  $| b_0 \wedge b_1 | b_0 \vee b_1$
- Com

  c::=cobegin c||c coend | skip | X:=a |c_0;c_1 |wait(b) |if b then c_0 else c_1|while b do c
- The range of variables integers of [0,2]
- Values of boolean variables: 0 and 1
- The name of all variables are single lower letter

experiment(cont’)

- Please translate the concurrent program into the first order logic.
- Please translate the first order logic into Kripke structure and draw a graph to represent KS.
- 3-part tool： Graphyviz
- official website： http://www.graphviz.org/ (better over wall)
- download link： http://soft.hao123.com/soft/appid/6971.html

summary

- Kripke Structure
- Translate one order logic into Kripke Structure
- Modeling digital circuit
- Modeling concurrent program