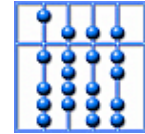


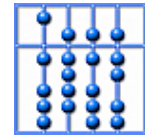
Specification of Distributed Systems

Dr. Bernhard Schätz
Leopold-Franzens Universität Innsbruck
Sommersemester 2005



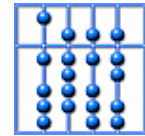
Overview

1. Introduction
2. Basics: Behavior, Interaction, Concurrency
3. Coroutines
4. Communicating Processes
5. Data Flow Models
6. State-Based Models
7. Coordination
8. Executions
9. Property Descriptions

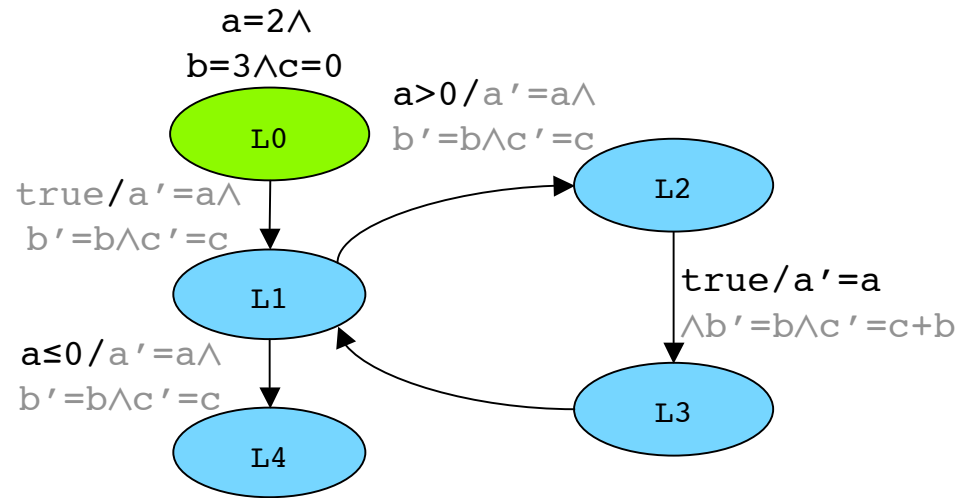


Overview

1. Introduction
2. Basics
3. Coroutines
 1. Modeling Complex Computation: Extended Transition Systems
 2. Model Implicit Interaction: Shared Memory Models
 3. Application: Threads
 4. Concepts: Interference, race conditions
4. Data Flow Models
5. Communicating Processes
6. Coordination
7. State-Based Models
8. Executions
9. Property Descriptions



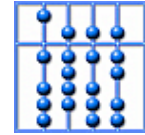
3.1 Summary: Modeling Complex Computations



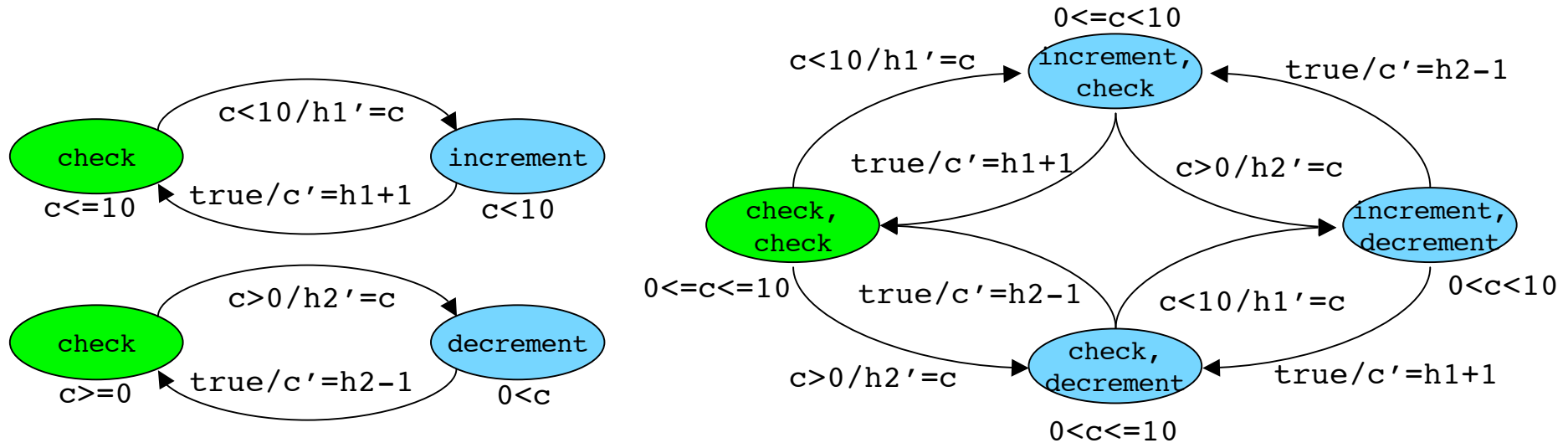
Concepts:

- State: Observation control and data aspects of a system at a specific instance of time
- Invariant: Collection of data states (of a control location)
- Extended Transition relation: Set of possible actions of a computation characterized by pre- and postcondition

Model: Extended Transition System (S, S_0, T)



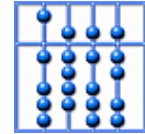
3.2 Summary: Modeling Shared Memory



Concepts:

- Shared Memory: Product space with joint global state and independent local and control state
- Implicit interaction: Interleaved execution of interactions

Model: Combined Extended Transition System



Example: Formalization

```
class InGate extends Thread {
  ...
  Counter capacity;

  InGate(Counter c)
  { capacity = c; }

  public void run() {
    ...
    while (true){
      ...
      capacity.decrement();
      ...
    }
  }
}
```

```
class Counter {
  int amount = 0;

  Counter(int a)
  { amount = a; }

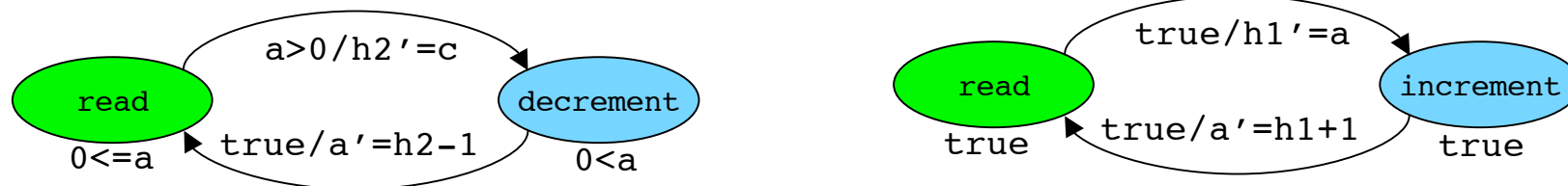
  void increment() {
    this.amount = this.amount + 1;
  }

  void decrement() {
    if(this.amount > 0)
      this.amount = this.amount -
1;
  }
}
```

```
class OutGate extends Thread {
  ...
  Counter capacity;

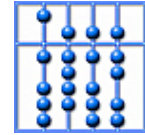
  OutGate(Counter c)
  { capacity = c; }

  public void run() {
    ...
    while (true){
      ...
      capacity.increment();
      ...
    }
  }
}
```

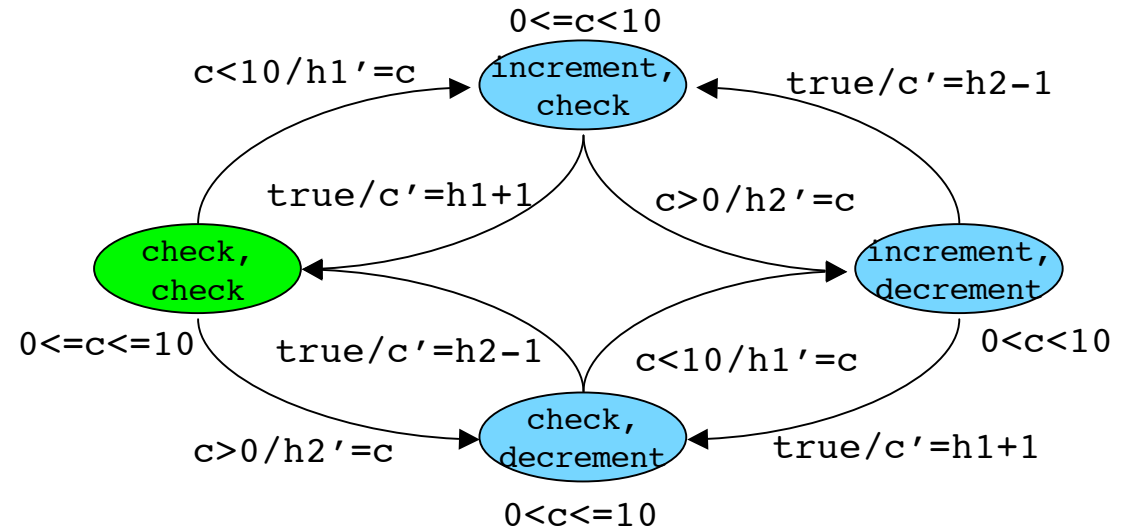
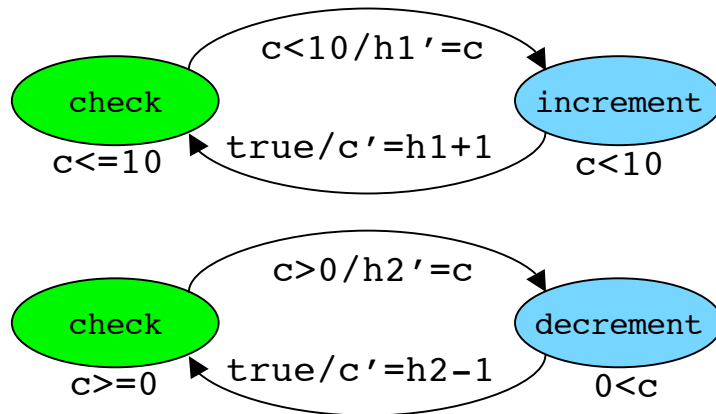


Java Virtual Machine

- Threads correspond to unsynchronized processes
- Shared objects correspond to shared variables, method variables to local variables
- Addition/subtraction of class variables are not atomic actions



3.4 Summary: Problems of Co-Routines

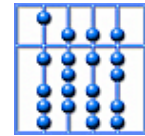


Issues:

- Interference: Observations of a co-routine in isolation do no longer hold when co-routine is composed with interfering co-routine
- Race condition: Observations of co-routines do no longer hold hold when re-ordering internal actions of the system

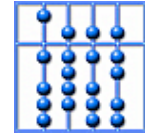
Consequence: Co-routines are not suitable concerning

- Modular development
- Platform-independent development



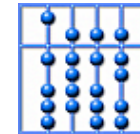
3.5 Questions

1. Model a shared buffer with limited capacity.
2. What are the execution traces of two routines using a shared bounded counter variable (initialized by 0)? Describe the traces of one process incrementing the variable as well as the execution traces of the other process decrementing the variable?
3. What is the (external) interface of each of the co-routines?
4. What are the observation traces (reduced to the common interface of the co-routines) of each co-routine?
5. What is the difference between the observation traces of a synchronized labeled transition system and an extended transition system?
6. What are the execution traces of the combined co-routines? What are the observation traces of the combined co-routines?
7. Can the execution/observation traces of the combined system be obtained from the execution/observation traces from the co-routines in isolation?
8. What are the execution/observation traces obtained if the technique to generate combined traces is applied, which is used for SLTS?
9. What kind of interaction is reflected in this kind of combined of the co-routines?



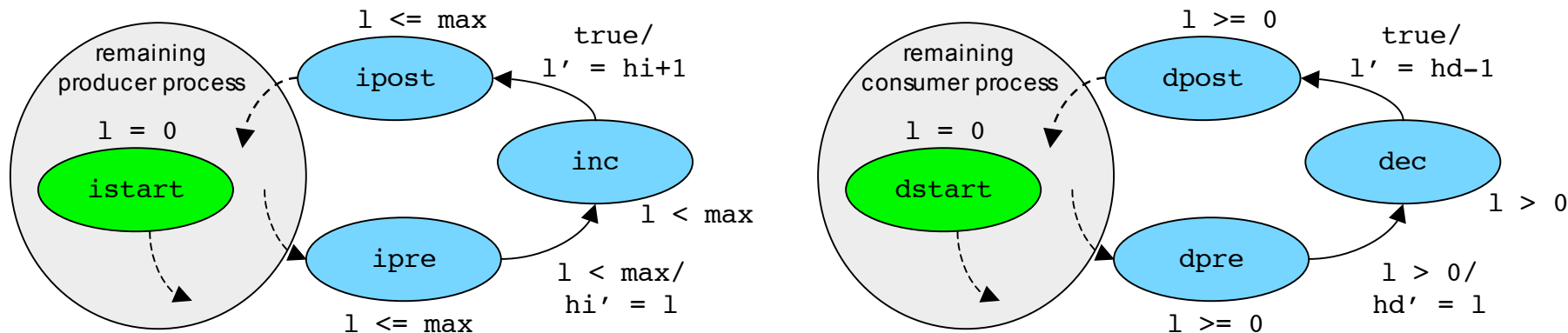
Questions:

10. How must the execution/observation traces of each co-routine be extended to support a shared-variable communication?
11. How must the ETS-models of each co-routine be adapted to reflect the extension of the execution/observation traces?
12. What is the relation between the issue of interference and the adapted ETS?
13. What are the differences between the two adapted ETS and SLTS?
14. Looking at the adapted ETS, what is the difference between a shared variable and a local variable of an ETS? What is the result for an interface description of a co-routine?
15. What is the model of a integrator routine reading the value of a (shared input) variable in, and adding it to a (shared output) variable acc as its output variable; furthermore, it resets acc to 0 if its (shared input) variable reset is not equal to 0?
16. Is there a difference between a shared, an input, and an output variable of a co-routine? How would this reflect in the traces and ETS following the approach in Questions 10 and 11?
17. If the integrator is embedded into two routines, one process producing values to be integrated, one reading the integrated value. Does the combined system behave as intended?
18. What are two important ingredients of concurrent interaction?



Question 2

What are the execution traces of two routines using a shared bounded counter variable (initialized by 0)?



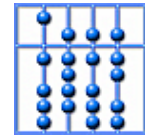
In a co-routine system, the counter structure cannot be modeled explicitly as a separate process. It is modeled as part of the data space; the control part is modeled as part of the consumer and the producer.

Incrementing routine: Contains transition increasing the shared variable

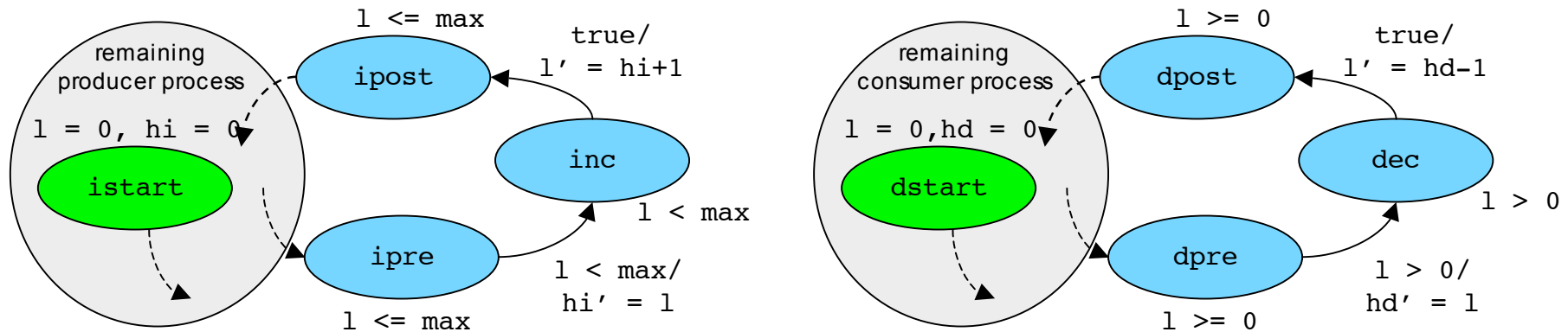
- $S \supset \{ (ipre, l, hi, \dots) \mid l < \max \} \cup \{ (inc, l, hi, \dots) \mid l = hi < \max \} \cup \{ (inc, l, hi, \dots) \mid hi + 1 = l \leq \max \}$
- $T \supset \{ ((ipre, l, hi, \dots), (inc, l, hi, \dots)) \mid l, hi < \max \} \cup \{ ((inc, l, hi, \dots), (ipost, l+1, hi, \dots)) \mid l, hi < \max \}$

Decrementing routine: Contains transition decreasing the shared variable

- $S \supset \{ (dpre, l, hd, \dots) \mid l > 0 \} \cup \{ (dec, l, hd, \dots) \mid l = hd > 0 \} \cup \{ (dpost, l, hd, \dots) \mid hd = l - 1 \geq 0 \}$
- $T \supset \{ ((dpre, l, hd, \dots), (dec, l, hd, \dots)) \mid l, hd > 0 \} \cup \{ ((dec, l, hd, \dots), (dpost, l+1, hd, \dots)) \mid l, hd > 0 \}$

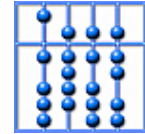


Question 2



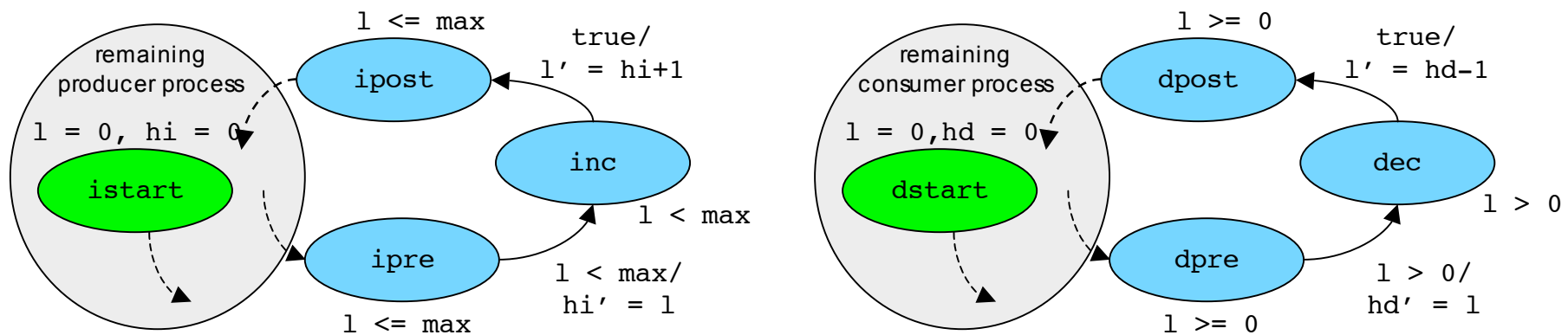
Execution traces changing shared variable (assuming $\max = 10$):

- Incrementing routine:
(istart,0,0,...)•...•(ipre,0,...)•(inc,0,0...)•(ipost,1,0...)•...•(ipre,1,...)•(inc,1,1...)•
(ipost,2,1...)•...•(ipre,9,...)•(inc,9,9...)•(ipost,10,9...)•... •(ipre,10,...)
- Decrementing routine:
(dstart,0,0,...)•...•(dpre,0,...)



Question 3

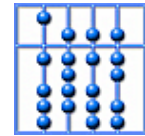
What is the (external) interface of each of the co-routines?



Interface: Part of a system that is either observed or influenced by the environment of the system

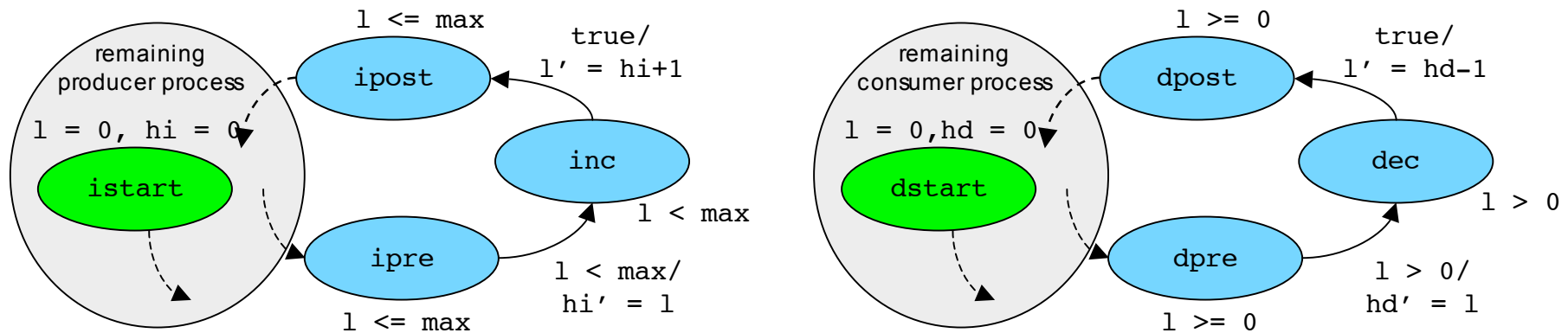
Co-routines do not have an explicit interface, however

- Producer: Shared variable l observed and changed by consumer
- Consumer: Shared variable l observed and changed by producer



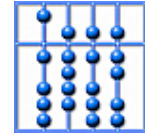
Question 4

What are the observation traces (reduced to the common interface of the co-routines) of each co-routine?



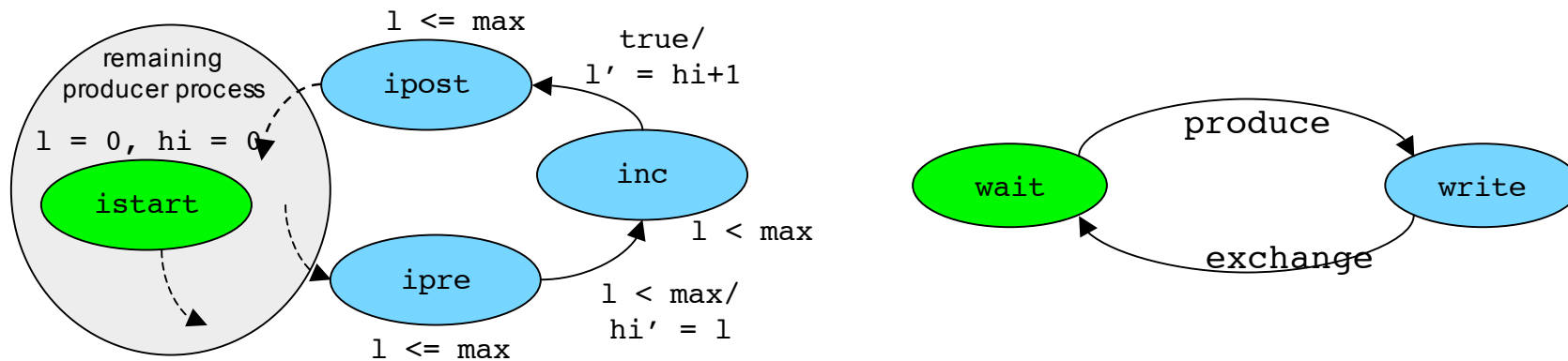
Observation traces changing shared variable (assuming $\max = 10$):

- Incrementing routine:
 $(0) \cdot (0) \cdot \dots \cdot (1) \cdot (1) \cdot \dots \cdot (2) \cdot (2) \cdot \dots \cdot (10)$
- Decrementing routine:
 $(0) \cdot \dots \cdot (0)$



Question 5

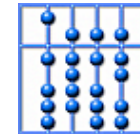
What is the difference between the observation traces of a synchronized labeled transition system and an extended transition system??



Observation traces:

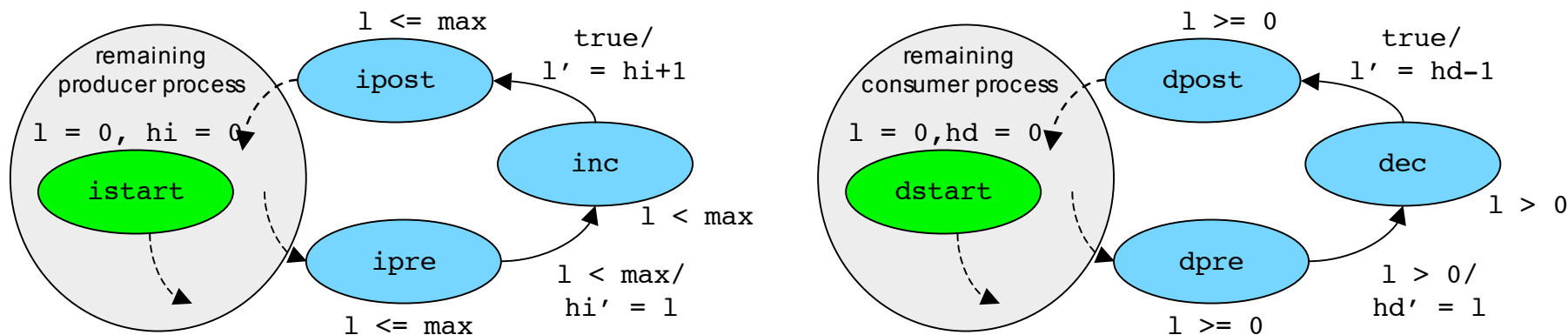
- Labeled Transition System:
Sequence of interactions (i.e., labels of transitions)
- Extended Transition System:
Sequence of states

Note: Sequence of states may contain “stuttering steps” with unchanged states



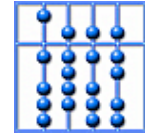
Question 6

What are the execution traces of the combined co-routines? What are the observation traces of the combined co-routines??



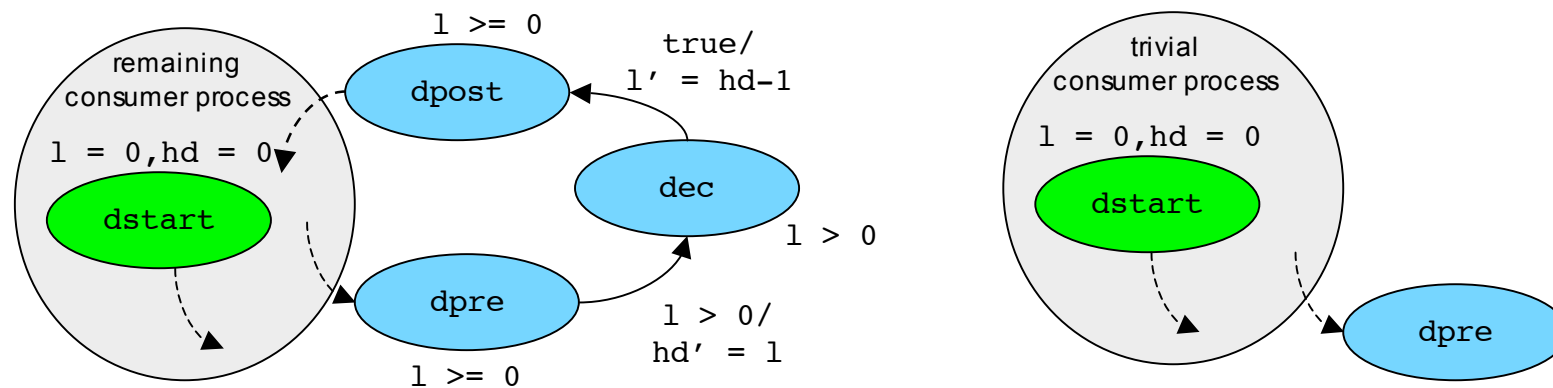
Combined system: Arbitrary interleaving of actions of producer and consumer

- Execution traces:
Traces over product state with shared variable l , e.g.,
(istart,dstart,0,0,0...) \bullet ... \bullet (ipre,dstart,0,..,0..) \bullet (inc,dstart,0,0,0...) \bullet ...
 \bullet (ipre,dpre,0,0,...)(ipost,dpre,1,0,0...) \bullet (ipost,dec,1,0,1,...) \bullet (ipost,dpost,0,0,1...)
- Observation traces:
Traces over values of l , e.g., (0) \bullet (0) \bullet ... \bullet (1) \bullet (1) \bullet ... \bullet (2) \bullet (2) \bullet ... \bullet (1) \bullet
(1) \bullet ... \bullet (2) \bullet (2) \bullet ... \bullet (3) \bullet (3) \bullet ... \bullet (10)



Question 7

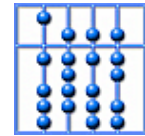
Can the execution/observation traces of the combined system be obtained from the execution/observation traces from the co-routines in isolation?



Hint: Consider substituting the consumer by the trivial process doing nothing.

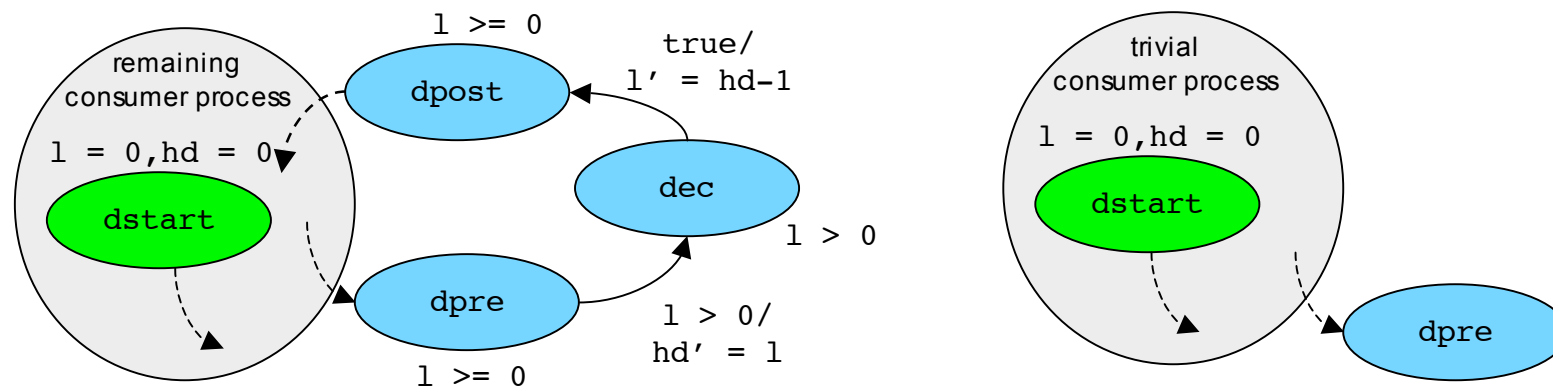
Consumer routine and trivial consumer routine have

- Identical execution traces (i.e., $(dstart, 0, 0, \dots) \bullet \dots \bullet (dpre, 0, \dots)$)
- Identical observation traces (i.e., $(0) \bullet \dots \bullet (0)$)



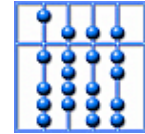
Question 7

Can the execution/observation traces of the combined system be obtained from the execution/observation traces from the co-routines in isolation?



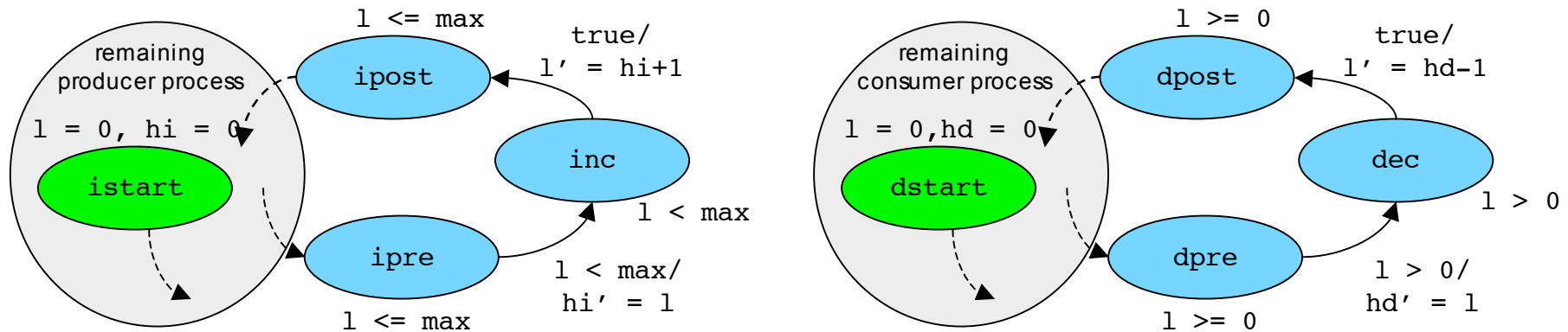
However, the execution/observation traces of the combined systems differ!

Therefore: The execution/observation traces of the combined system cannot be derived by looking only at the execution/observation traces of the co-routines in isolation!



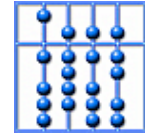
Question 8

What are the execution/observation traces obtained if the technique to generate combined traces is applied, which is used for SLTS?



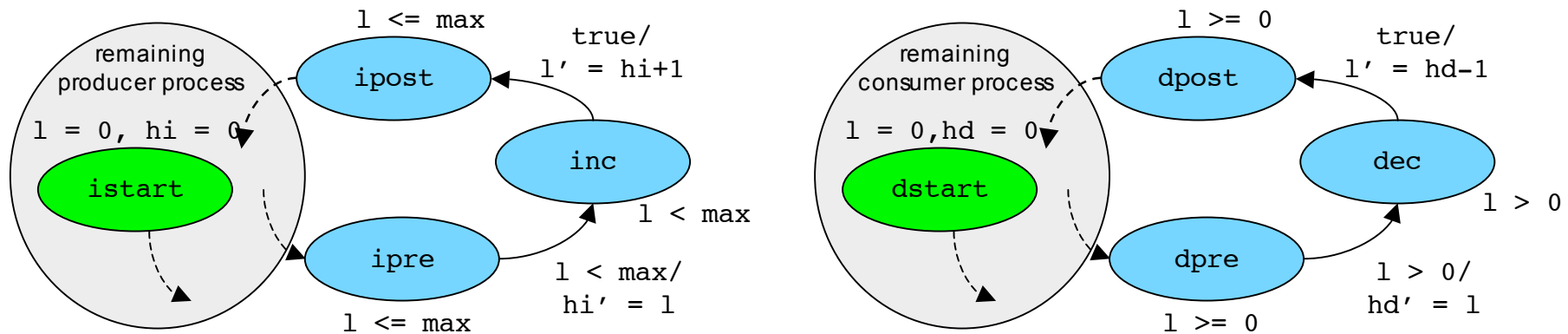
SLTS-based generation of traces: Traces of combined reduced to observation about one component must be traces of that component.

- Observation traces: Which combined traces are conforming with $(0) \bullet \dots \bullet (0)$ of the decrementing routine?
- Only subtrace $(0) \bullet \dots \bullet (0)$ of, e.g., $(0) \bullet (0) \bullet \dots \bullet (1) \bullet (1) \bullet \dots \bullet (2) \bullet (2) \bullet \dots \bullet (10)$



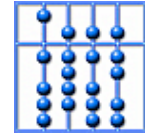
Question 8

What are the execution/observation traces obtained if the technique to generate combined traces is applied, which is used for SLTS?



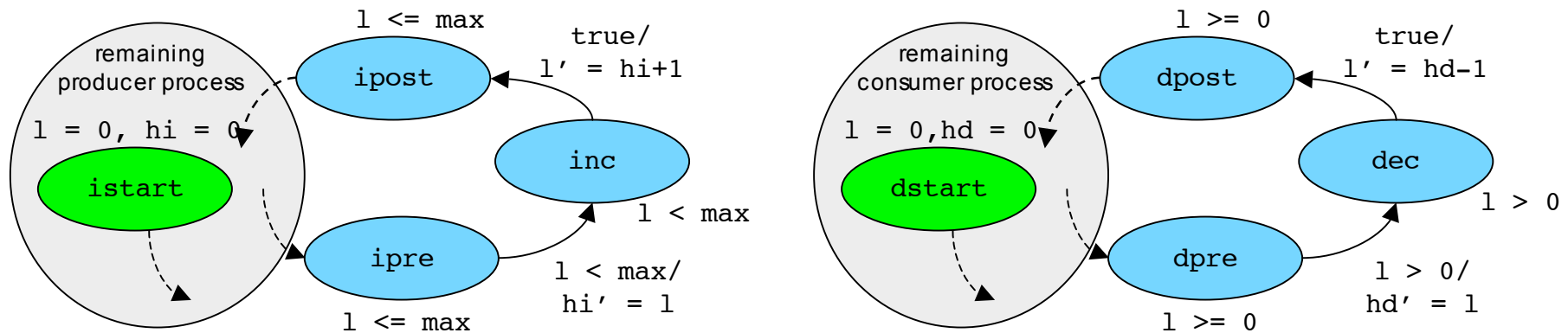
SLTS-based generation of traces: Traces of combined reduced to observation about one component must be traces of that component.

- Execution traces: Which combined traces are conforming with $(dstart, 0, 0, \dots) \cdot \dots \cdot (dpre, 0, \dots)$ of the decrementing routine?
- Example: $(istart, dstart, 0, 0, 0, \dots) \cdot \dots \cdot (ipre, dstart, 0, \dots, 0, \dots) \cdot (inc, dstart, 0, 0, 0, \dots) \cdot \dots \cdot (ipre, dpre, 0, \dots)$



Question 9

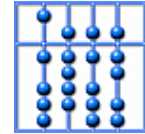
What kind of interaction is reflected in this kind of combined of the co-routines?



Observation:

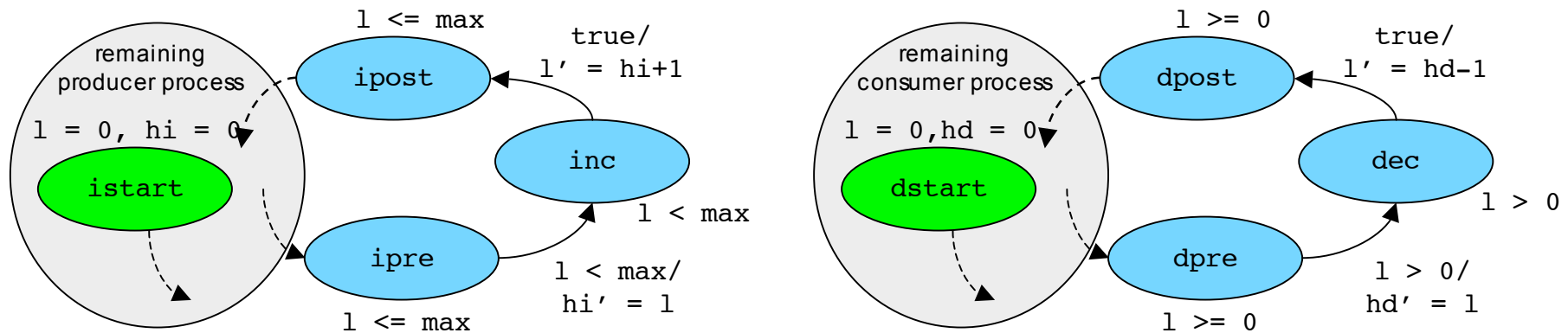
- Producer routine can only increment shared variable
- Consumer routine can only decrement shared variable

Blocking interaction: Producer and consumer do not agree on the change of the variable



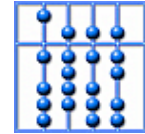
Question 10

How must the execution/observation traces of each co-routine be extended to support a shared-variable communication??



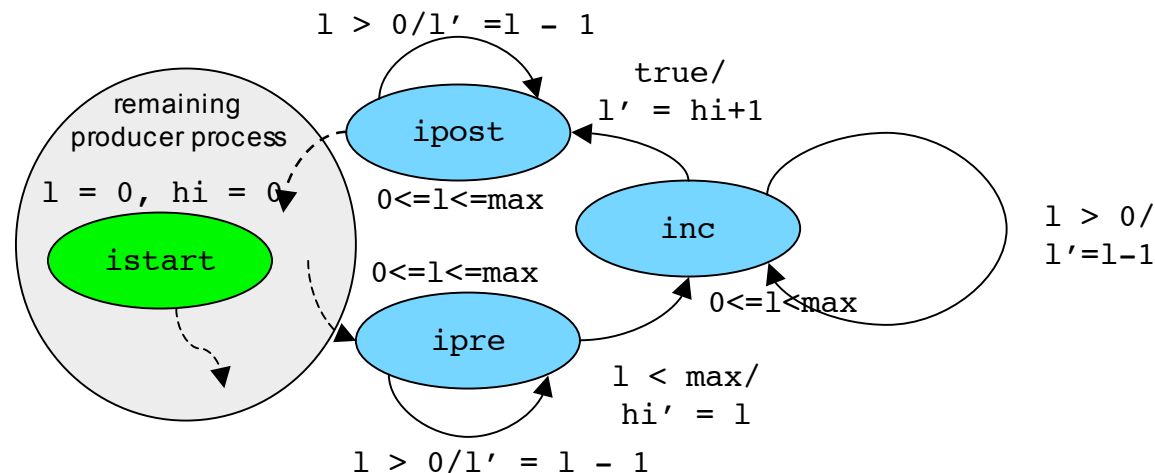
Execution/observation traces must allow to arbitrarily change the shared variable:

- Extending $(dstart, 0, 0, \dots) \cdot \dots \cdot (dpre, 0, \dots)$ to $(dstart, 0, 0, \dots) \cdot \dots \cdot (dpre, 0, \dots) \cdot \dots \cdot (dpre, 1, \dots)$
- Allows to combine $(istart, dstart, 0, 0, 0, \dots) \cdot \dots \cdot (ipre, dstart, 0, \dots, 0, \dots) \cdot (inc, dstart, 0, 0, 0, \dots) \cdot \dots \cdot (inc, dpre, 0, 0, \dots) \cdot (ipost, dpre, 1, 0, 0, \dots)$



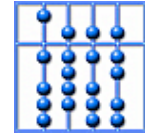
Question 11

How must the ETS-models of each co-routine be adapted to reflect the extension of the observation traces?



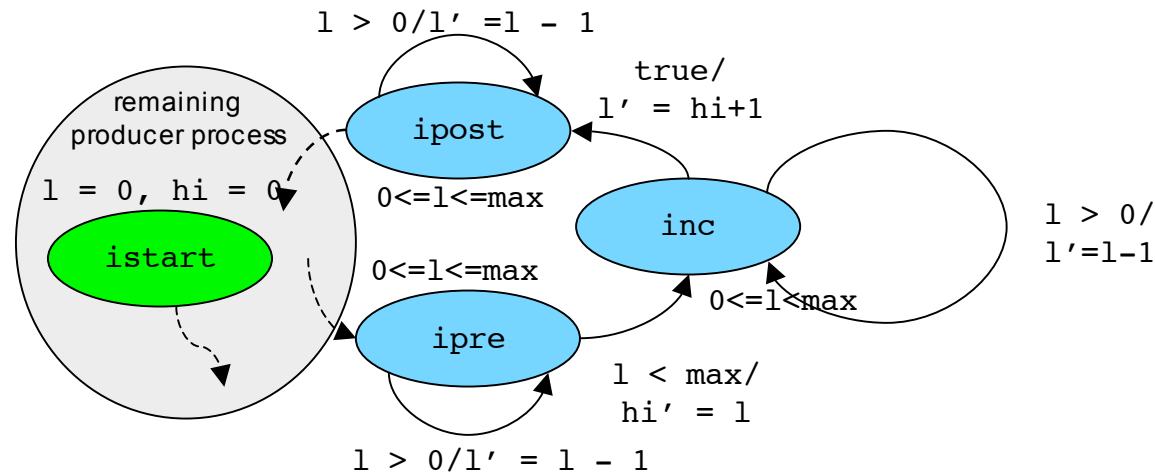
Adapting the extended transition system for observation traces:

- Adding a feedback loop to each state allowing to arbitrarily decrease (increase) the value of the shared variable l :
 $l > 0 / l' = l - 1$
- Execution traces: Additionally, adding stuttering transitions



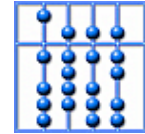
Question 12

What is the relation between the issue of interference and the adapted ETS?



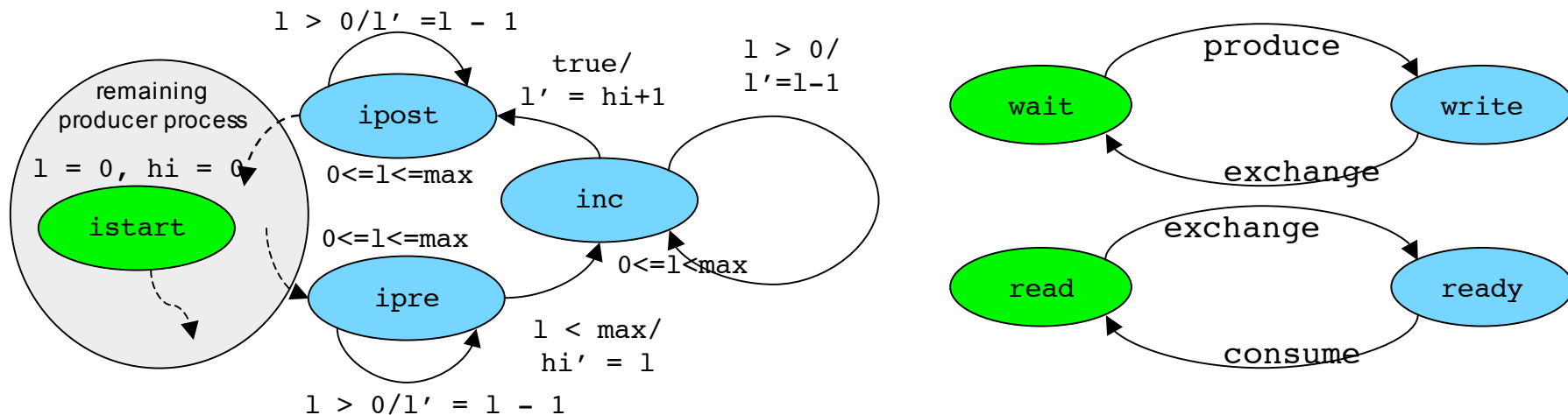
Adapting the extended transition system for observation traces:

- Feedback loop: Interaction of environment, arbitrarily decreasing (increasing) the value of the shared variable l
- Observations about original transition system do not hold about adapted transition system, e.g., value of l increases between $ipre$ and $ipost$



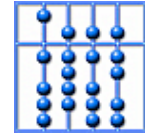
Question 13

What are the differences between the two adapted ETS and and an combination of two SLTS?



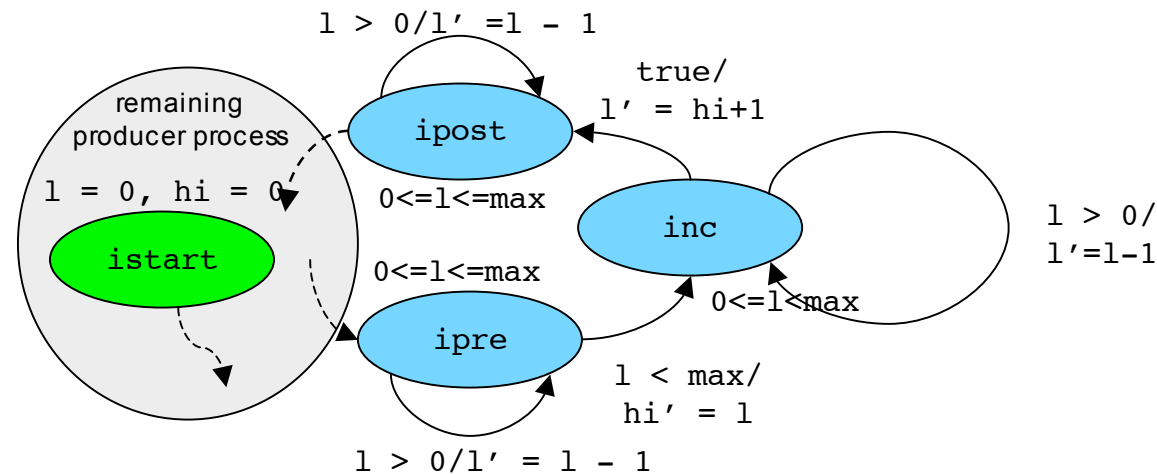
Both systems use synchronized transitions to reflect interaction:

- SLTS: shared interaction labels (e.g., exchange)
- ETS: common transitions for changes of shared variable (e.g., $l' = l - 1$ interacting with $hd = l / l' = hd - 1$)

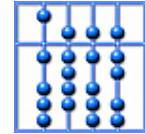


Question 14

Looking at the adapted ETS, what is the difference between a shared variable and a local variable of an ETS? What is the result for an interface description of a co-routine?

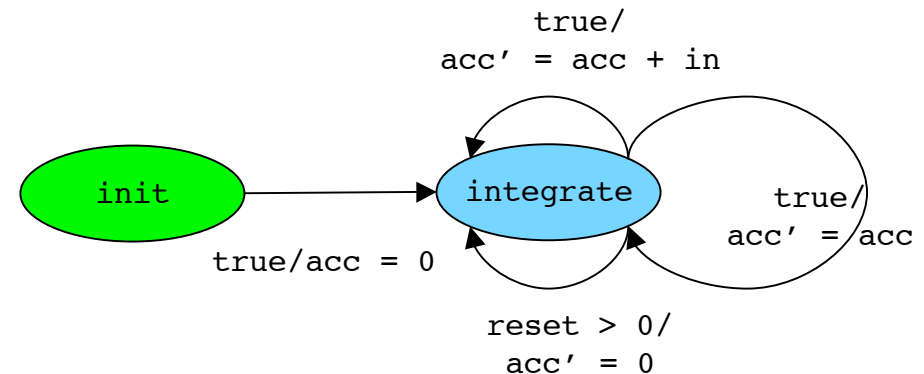
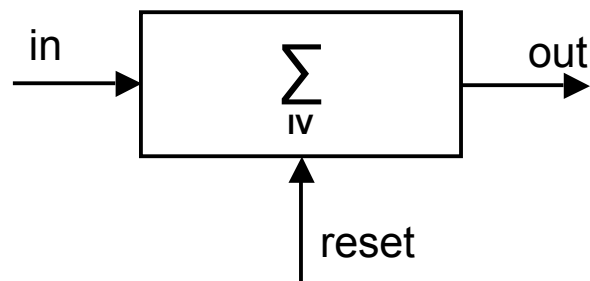


- Comparing shared/local variables: Shared variables are changed by the adapted ETS, local variables remain unchanged
- Interface description: In general, the data space should be separated into variables being changed arbitrarily by the environment (interface), and local variables, only changed by the routine



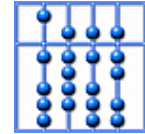
Question 15

What is the model of a integrator routine reading the value of a (shared input) variable in , and adding it to a (shared output) variable acc as its output variable;?



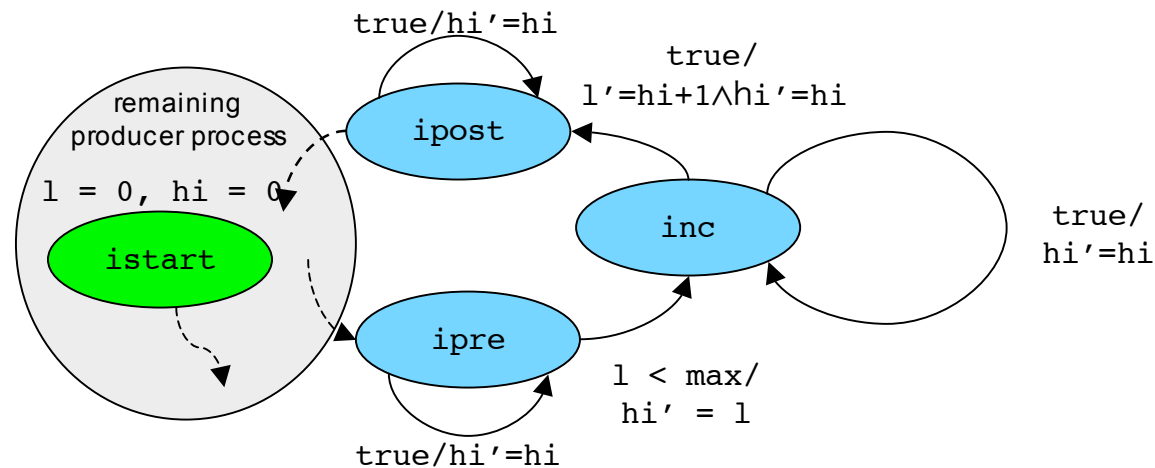
Actions:

- Adding input ($acc' = acc + in$)
- Resetting acc ($acc' = 0$)
- Stuttering step ($acc' = acc$)



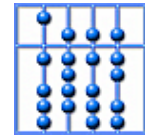
Question 16

Is there a difference between a shared, an input, and an output variable of a co-routine? How would this reflect in the traces and ETS following the approach in Questions 10 and 11?



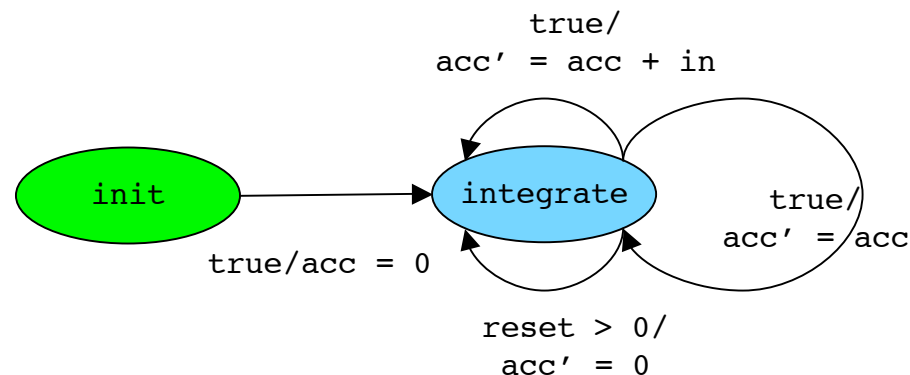
Interface description:

- Input changed arbitrarily by the environment, no assumption about values
- Local, output variables only changed by the routine



Question 17

What If the integrator is embedded into two routines, one process producing values to be integrated, one reading the integrated value. Does the combined system behave as intended?



- Hint: race conditions
- Problem: Information may be lost:
 - A possible input is lost if the sender is faster
 - A possible output is missed if the integrator is faster