Phd course on

Formal modelling and analysis of interactive systems

Part 2
Formal Tools and HCI Concepts

CSP Process Algebra, HCI Concepts and Modelling Human Behaviour

Antonio Cerone

United Nations University

International Institute for Software Technology

Macau SAR China

email: antonio@iist.unu.edu

web: www.iist.unu.edu

Contents

Formal Tools

Formal Methods

Traditional
Mathematical Modelling

Formal Methods

Formal Methods

Traditional
Mathematical Modelling

|

detailed ⟵ model

Formal Methods

Formal Methods

Traditional
Mathematical Modelling

detailed ← model

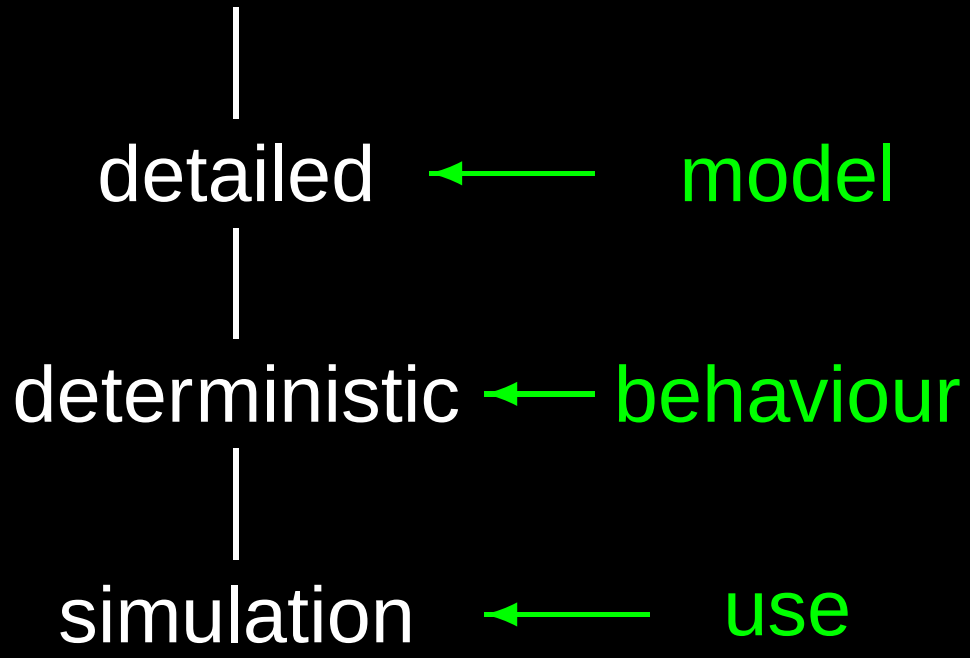
deterministic ← behaviour

Formal Methods

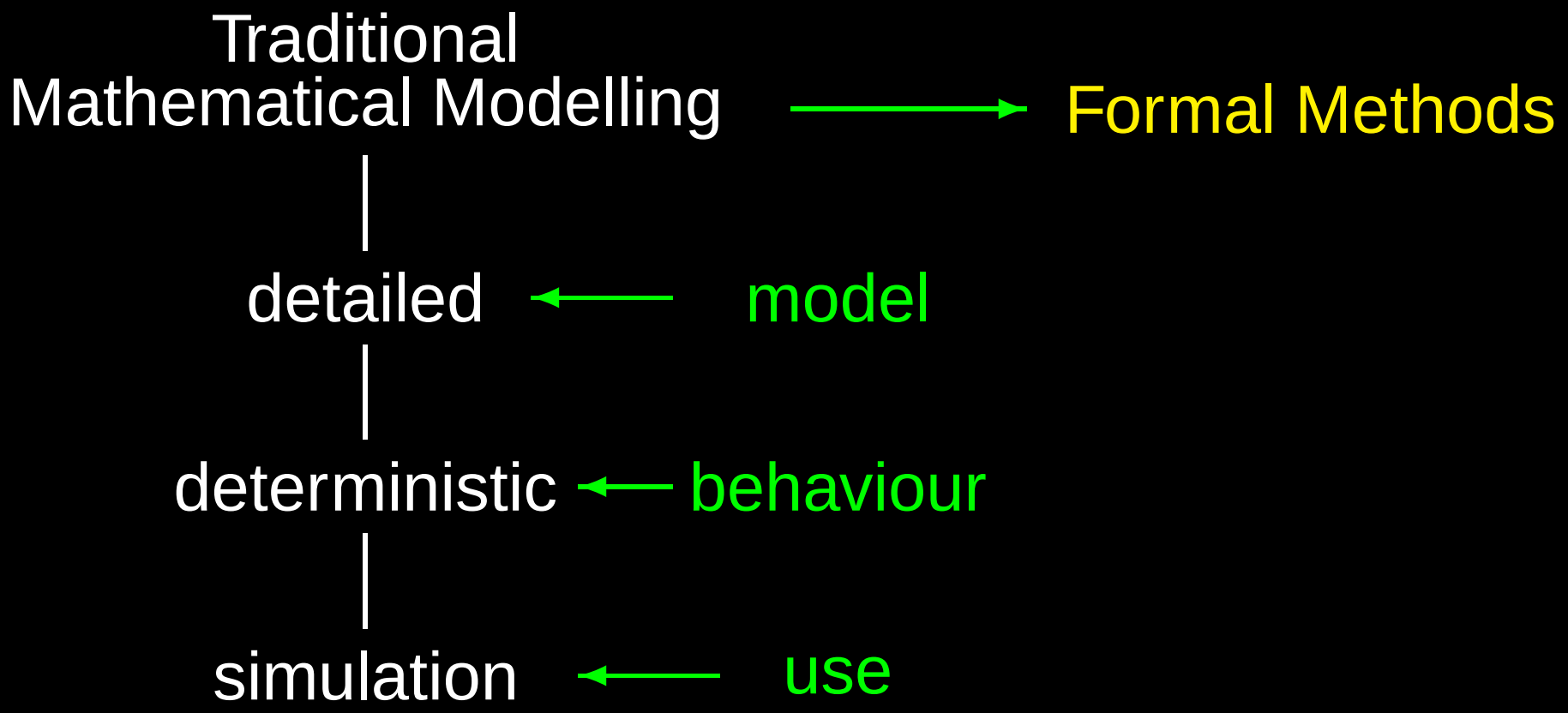
Formal Methods

Traditional Mathematical Modelling

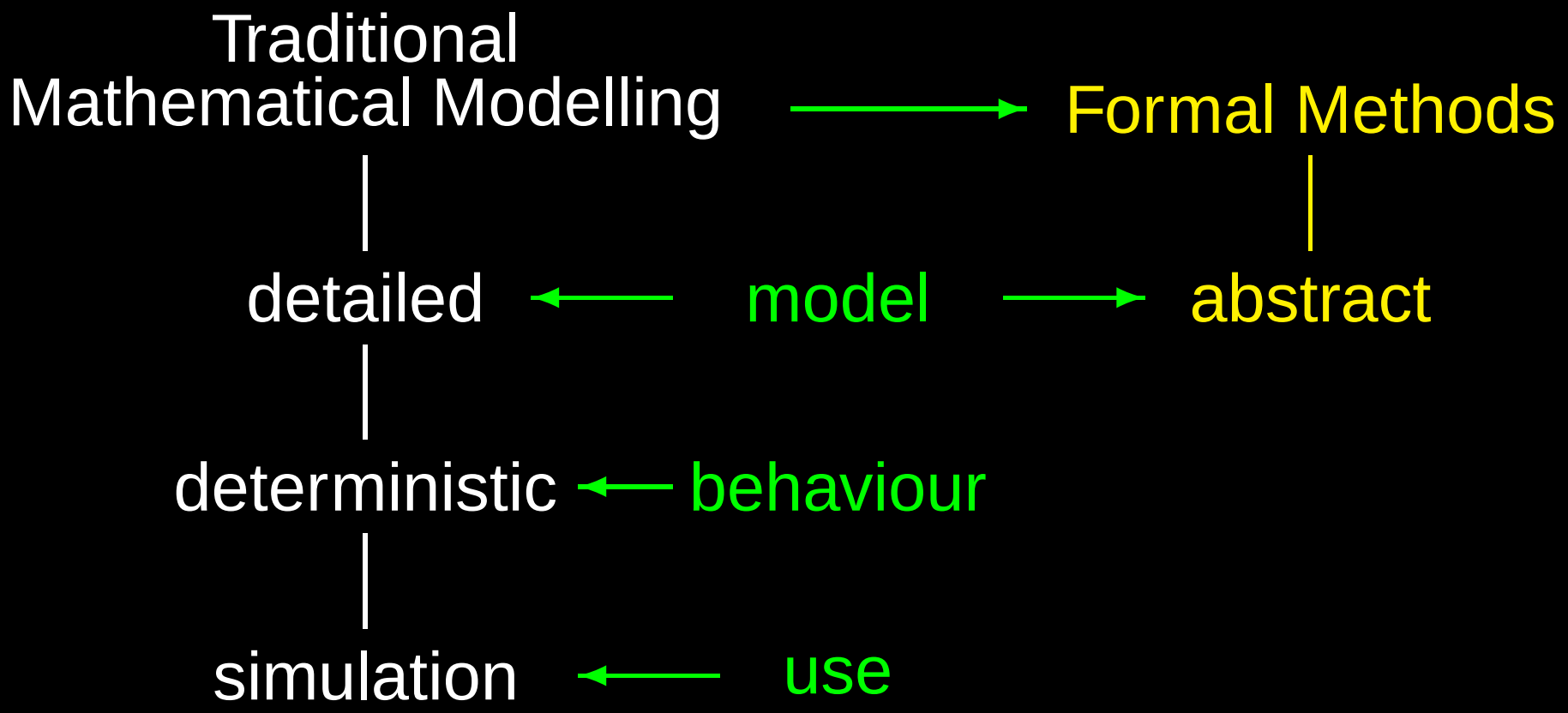
Formal Methods



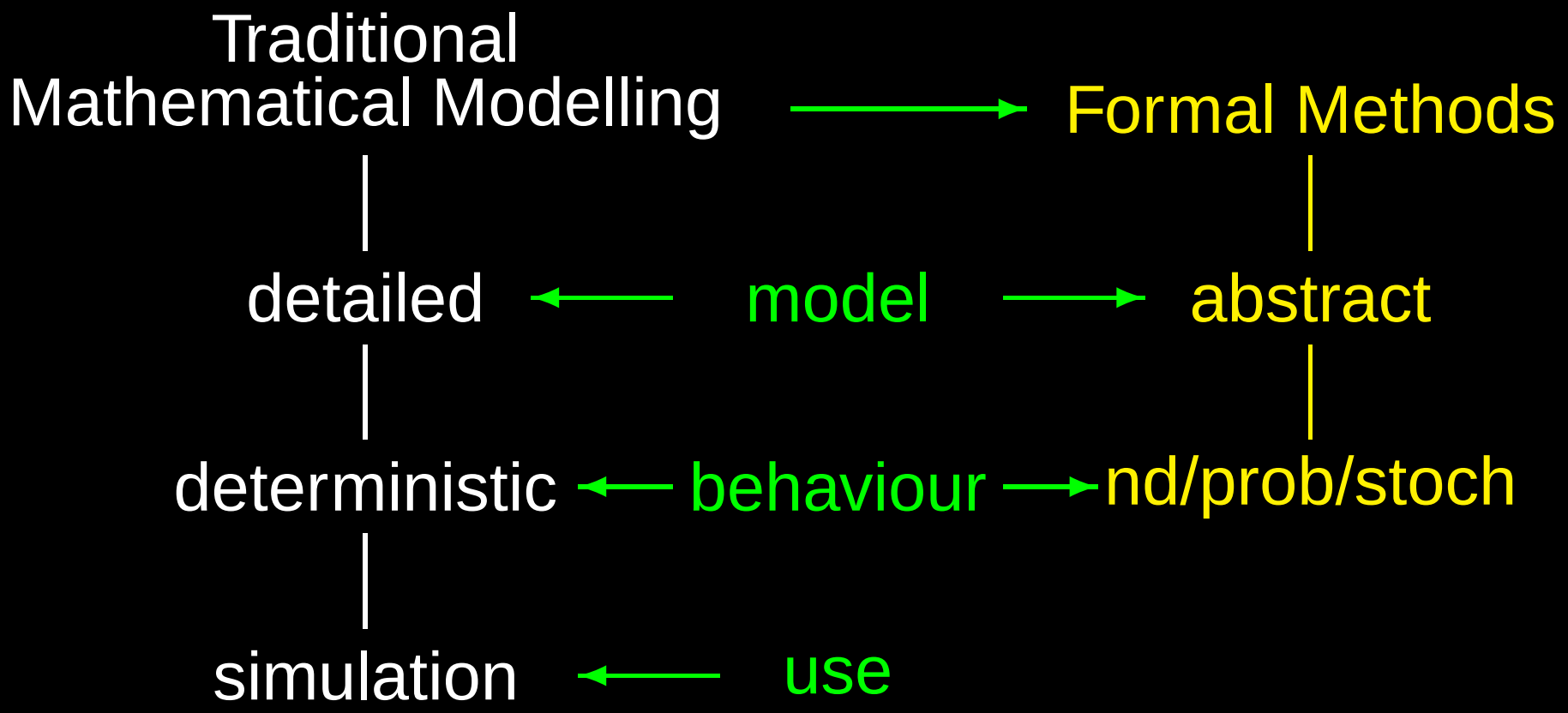
Formal Methods



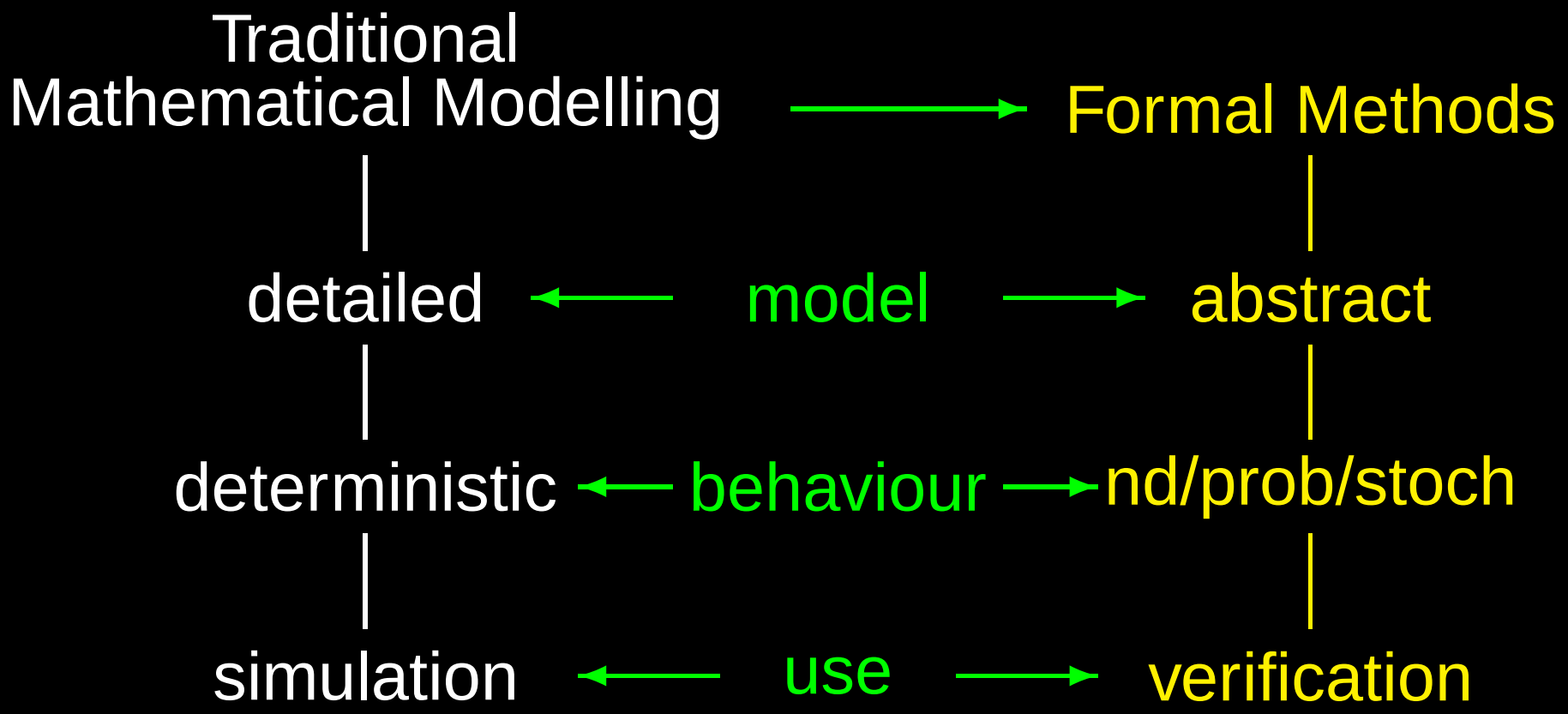
Formal Methods



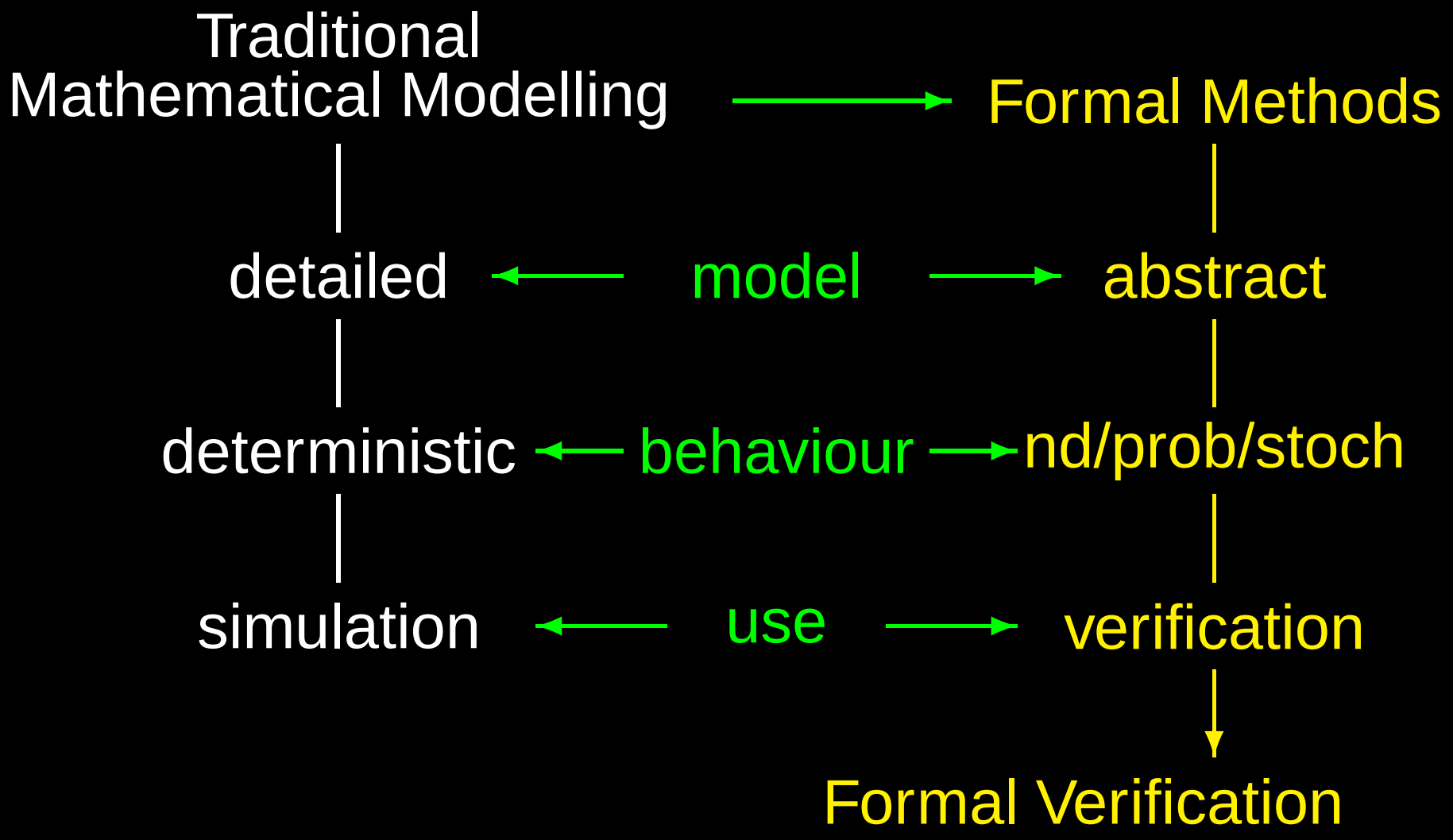
Formal Methods



Formal Methods



Formal Methods

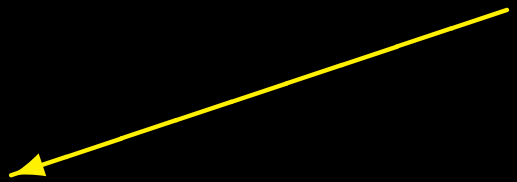


Formality

Formal Specification Language

Formality

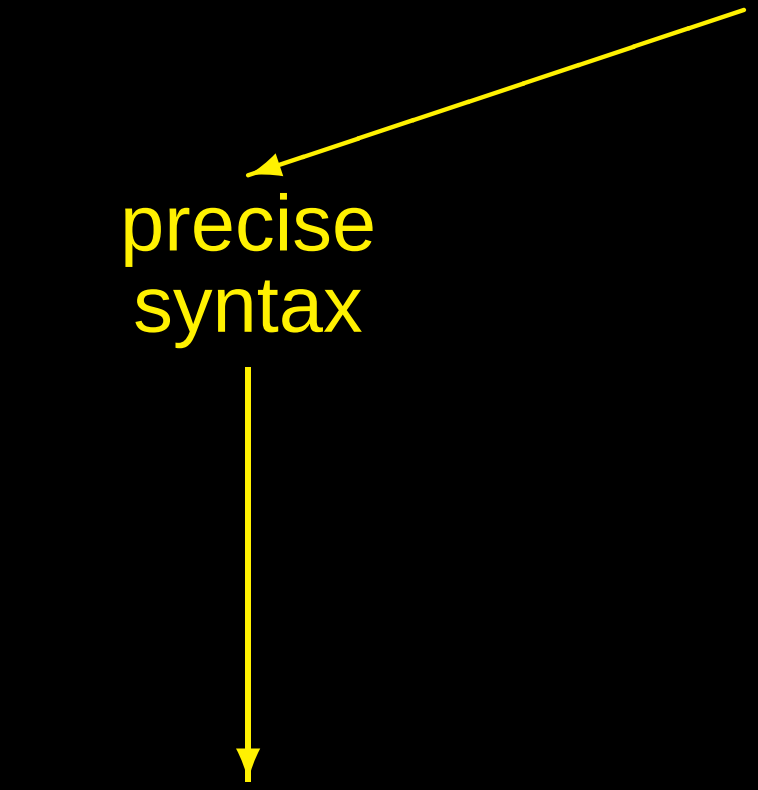
Formal Specification Language



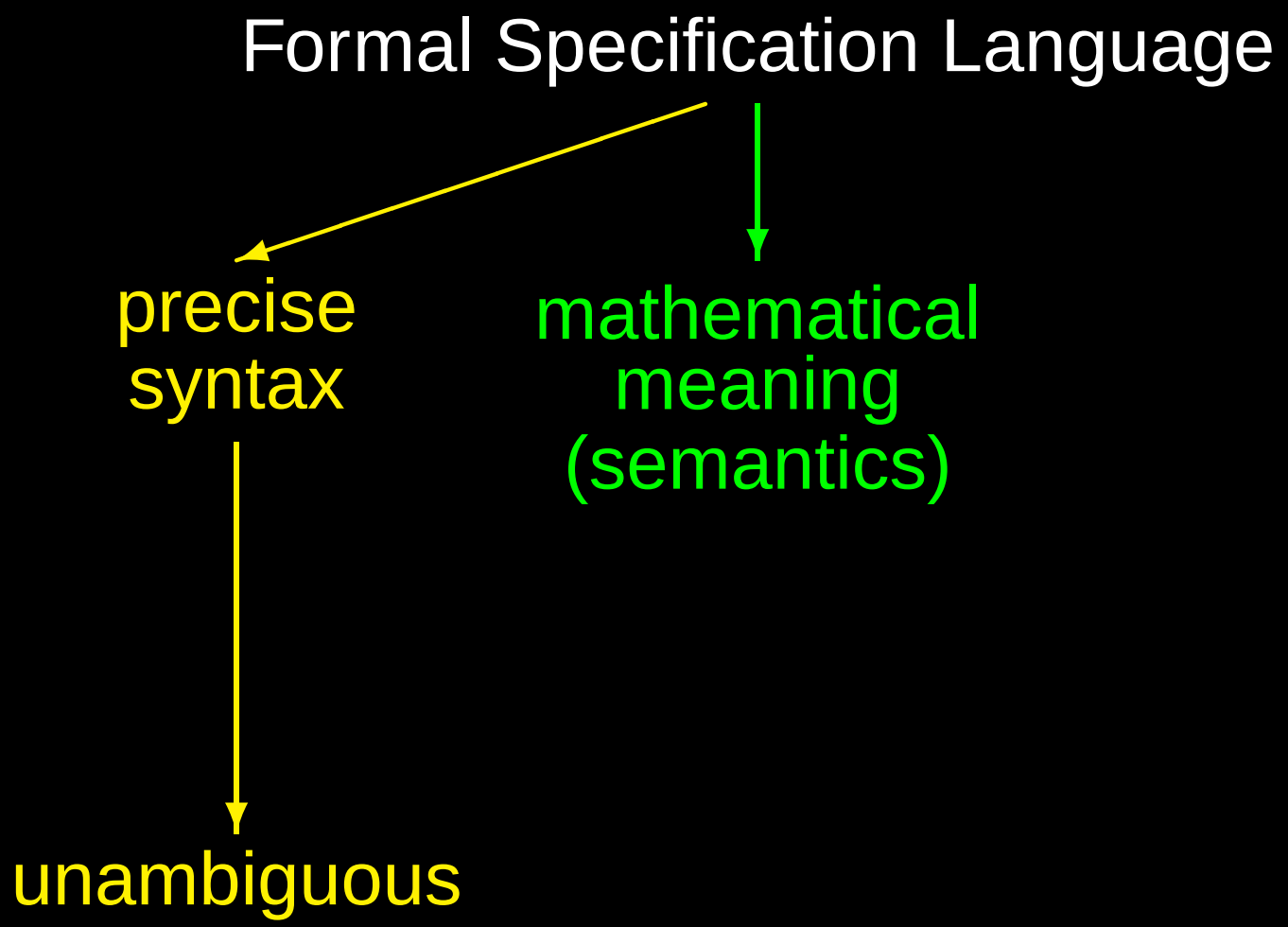
precise
syntax

Formality

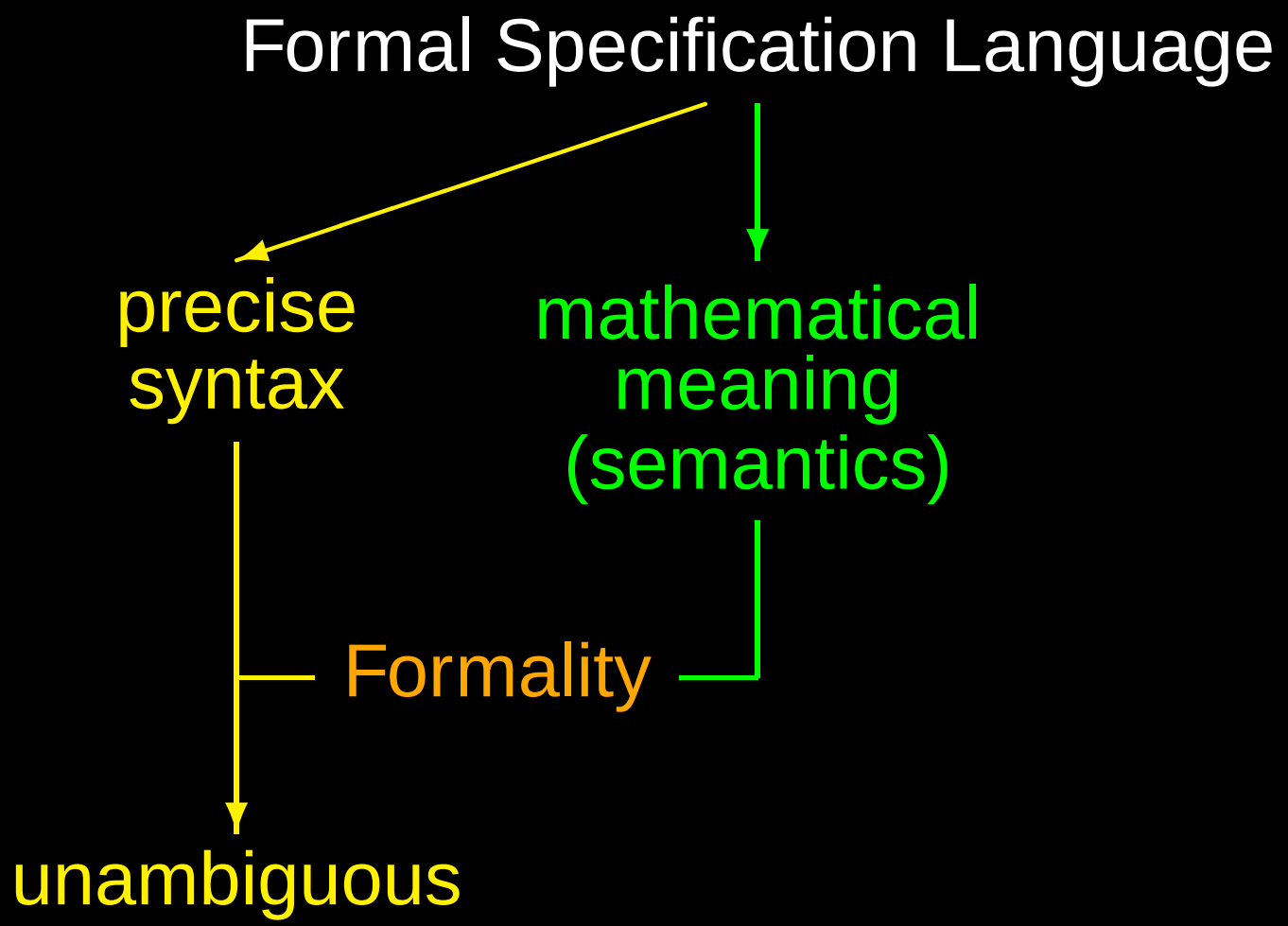
Formal Specification Language



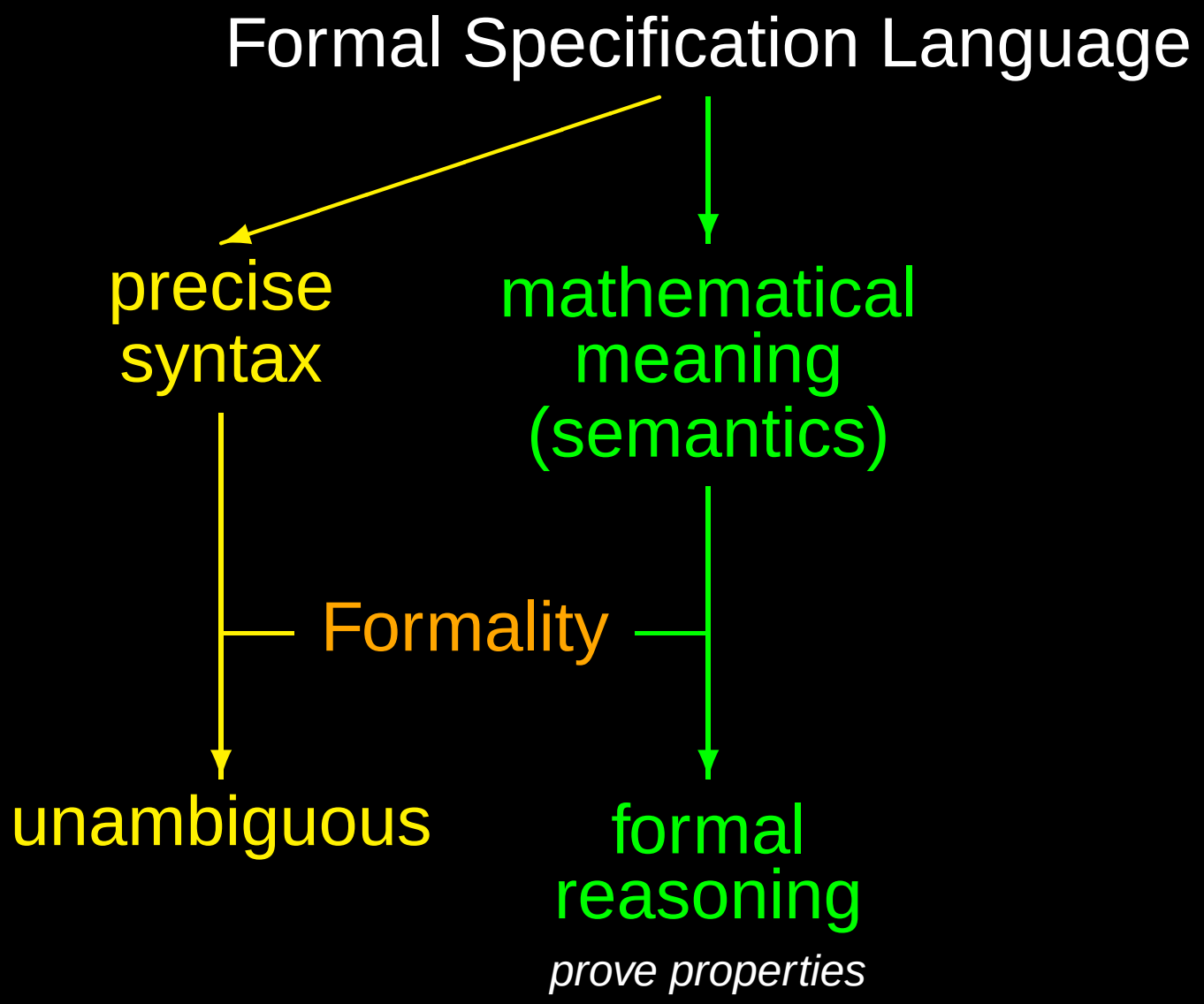
Formality


Formal Specification Language
precise syntax
mathematical meaning (semantics)
unambiguous

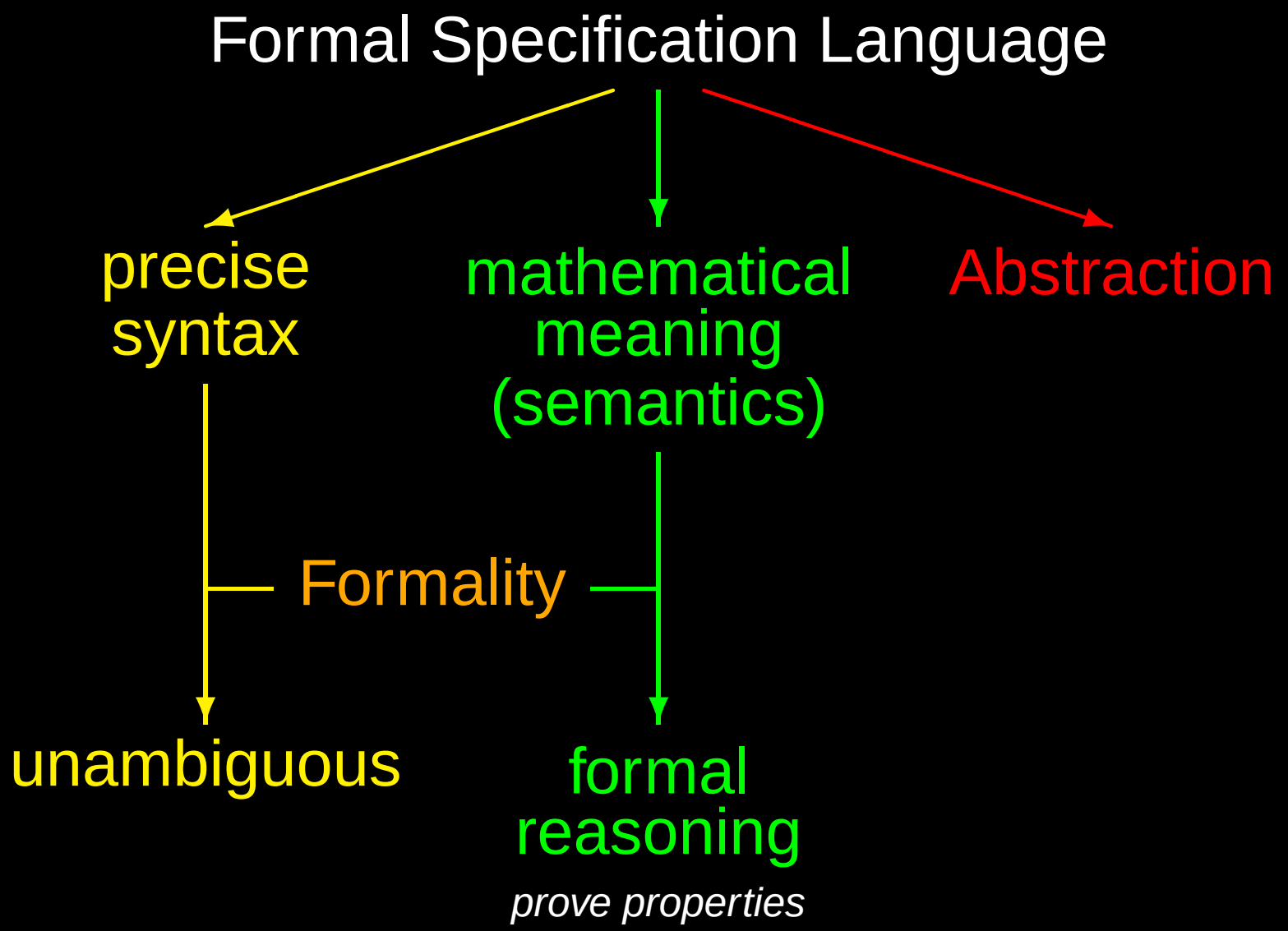
Formality


Formal Specification Language
precise syntax
mathematical meaning (semantics)
Formality
unambiguous

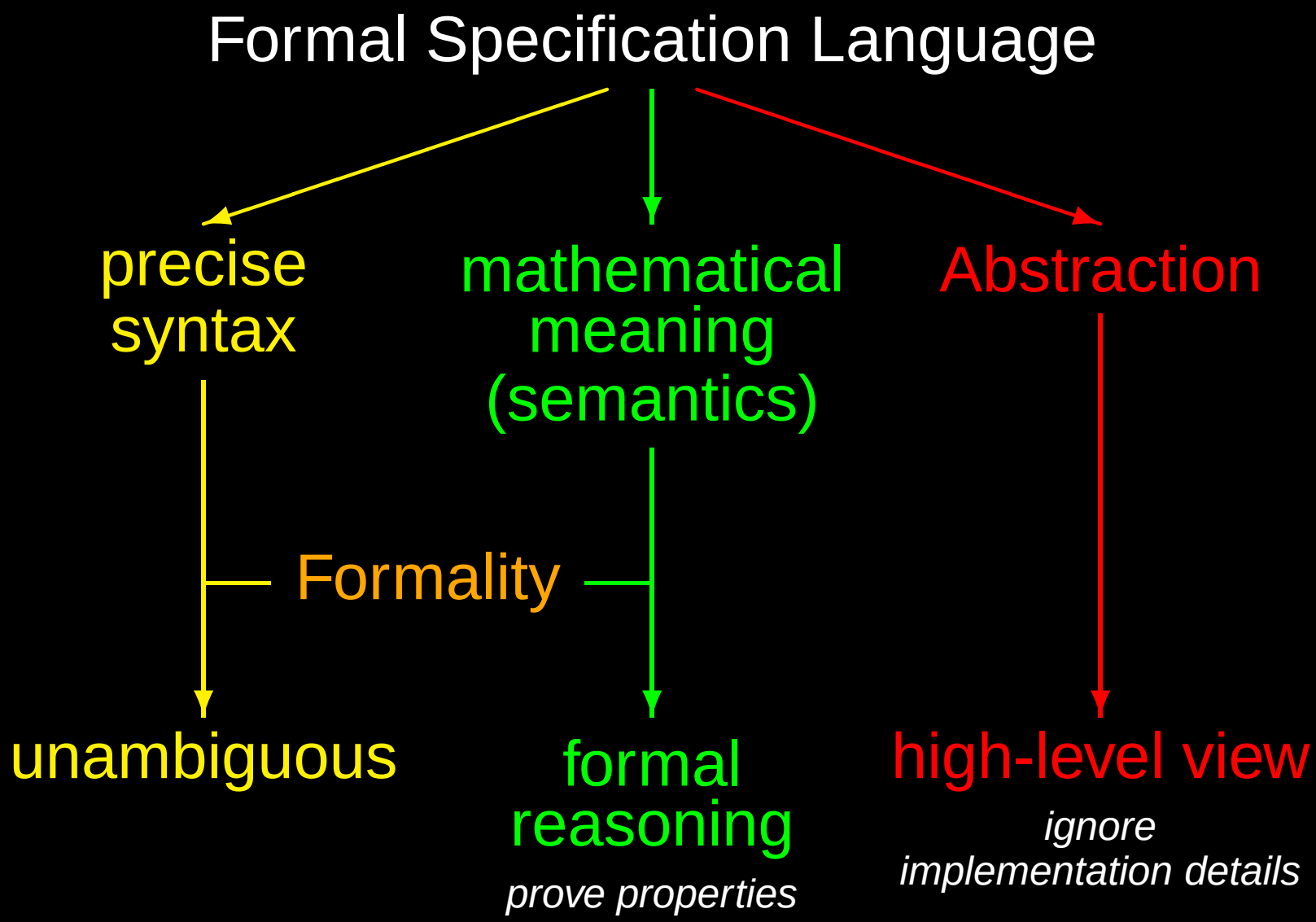
Formality


Formal Specification Language
precise syntax
mathematical meaning (semantics)
Formality
unambiguous
formal reasoning
prove properties

Formality


Formal Specification Language
precise syntax
mathematical meaning (semantics)
Abstraction
Formality
unambiguous
formal reasoning
prove properties

Formality


Formal Specification Language
precise syntax
mathematical meaning (semantics)
Abstraction
Formality
unambiguous
formal reasoning
prove properties
high-level view
ignore implementation details

Process Algebras

- event-based formal specification languages

Process Algebras

- event-based formal specification languages
- basic entities: actions and processes

Process Algebras

- event-based formal specification languages
- basic entities: actions and processes
- processes

Process Algebras

- event-based formal specification languages
- basic entities: actions and processes
- processes
  - evolve by performing actions

Process Algebras

- event-based formal specification languages
- basic entities: actions and processes
- processes
  - evolve by performing actions
  - are composed using operators

Examples of Process Algebras

- CSP — Communicating Sequential Processes
- CCS — Calculus of Communicating Systems
- CirCal — Circuit Calculus
- Lotos — Language of Temporal Ordering Specs
- ACP — Algebra of Communicating Processes

Examples of Process Algebras

CSP — Communicating Sequential Processes
CCS — Calculus of Communicating Systems
CirCal — Circuit Calculus
Lotos — Language of Temporal Ordering Specs
ACP — Algebra of Communicating Processes

Concurrency Workbench

The Concurrency Workbench of New Century (CWB-NC) supports

Concurrency Workbench

The Concurrency Workbench of New Century (CWB-NC) supports

- modelling using several process algebras: CCS and some extensions, Lotos, CSP
- simulation
- model-checking

Starting CWB-NC

```
shell>
```

Starting CWB-NC

```
shell> cwb-nc csp

The Concurrency Workbench of the New Ce
(Version 1.2 --- June, 2000)

cwn-nc>
```

Starting CWB-NC

```
shell> cwb-nc csp

The Concurrency Workbench of the New Ce
(Version 1.2 --- June, 2000)

cwn-nc> load atm-machine.csp
Execution time (user,system,gc,real):(0
cwn-nc>
```

Starting CWB-NC

```
shell> cwb-nc csp

The Concurrency Workbench of the New Ce
(Version 1.2 --- June, 2000)

cwn-nc> load atm-machine.csp
Execution time (user,system,gc,real):(0
cwn-nc> help

Available CWB-NC commands are:
   caching {on | off}
   cat identifier
   cd directory
   ...
```

Automatic Teller Machine (ATM) Example

Example: ATM Machine

Informal Specification

An ATM machine requires a user to

- insert a bank card;
- enter the right pin for that card

Example: ATM Machine

Informal Specification

An ATM machine requires a user to

- insert a bank card;
- enter the right pin for that card

Then the machine.

- delivers the cash to the user;
- returns the bank card to the user;
- waits that the user has collected cash and card before being ready for a new transaction.

Modelling

- extract from the informal description the actions that define the evolution of the system and label them with actions of the algebra

Modelling

- extract from the informal description the actions that define the evolution of the system and label them with actions of the algebra
- give a precise and unambigous interpretation to each identified action

Modelling

- extract from the informal description the actions that define the evolution of the system and label them with actions of the algebra
- give a precise and unambigous interpretation to each identified action, taking into account
  - which party (user or machine) performs the action
  - whether and how the other party interacts

Modelling

- extract from the informal description the actions that define the evolution of the system and label them with actions of the algebra
- give a precise and unambigous interpretation to each identified action, taking into account
  - which party (user or machine) performs the action
  - whether and how the other party interacts
- define the dynamic evolution of the system by applying the operators of the algebra to the actions

Example: ATM Machine

Informal Specification

An ATM machine requires a user to

- insert a bank card;
- enter the right pin for that card

Then the machine.

- delivers the cash to the user;
- returns the bank card to the user;
- waits that the user has collected cash and card before being ready for a new transaction.

Example: ATM Machine

Formalisation

An ATM machine requires a user to

- insert a bank card;
  action *card_in*, performed by user,
  machine "swallows" the card
- enter the right pin for that card

Then the machine.

- delivers the cash to the user;
- returns the bank card to the user;
- waits that the user has collected cash and card before being ready for a new transaction.

Example: ATM Machine

Formalisation

An ATM machine requires a user to

- insert a bank card;
- enter the right pin for that card
  action *pin*, performed by user,
  machine accepts the pin

Then the machine.

- delivers the cash to the user;
- returns the bank card to the user;
- waits that the user has collected cash and card before being ready for a new transaction.

Example: ATM Machine

Formalisation

An ATM machine requires a user to

- insert a bank card;
- enter the right pin for that card

Then the machine.

- delivers the cash to the user;
  action *cash_out*, performed by machine
- returns the bank card to the user;
- waits that the user has collected cash and card before being ready for a new transaction.

Example: ATM Machine

Formalisation

An ATM machine requires a user to

- insert a bank card;
- enter the right pin for that card

Then the machine.

- delivers the cash to the user;
- returns the bank card to the user;
  action *card_out*, performed by machine
- waits that the user has collected cash and card before being ready for a new transaction.

Example: ATM Machine

Formalisation

An ATM machine requires a user to

- insert a bank card;
- enter the right pin for that card

Then the machine.

- delivers the cash to the user;
- returns the bank card to the user;
- waits that the user has collected cash and card before being ready for a new transaction. actions *coll_cash*, *coll_card* performed by user detected by the machine

ATM Machine Actions

- *card_in*: user inserts card and machine swallows it
- *pin*: user enters pin and machine accepts it
- *cash_out*: machine delivers cash to user
- *card_out*: machine returns card to user
- *coll_cash*: user collects cash and machine detects it
- *coll_card*: user collects cash and machine detects it
- *ready*: machine is ready

CSP Prefix Operators

The Prefix Operator (in CWB-NC: `->`) defines the sequentialisation of two action, that is, in CSP terminology, that the first action prefixes the second.

CSP Prefix Operators

The Prefix Operator (in CWB-NC: `->`) defines the sequentialisation of two action, that is, in CSP terminology, that the first action prefixes the second.

Example:

```
card_in -> pin
```

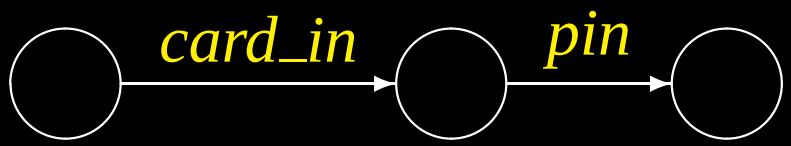
Action `card_in` prefixes action `pin`

Example: ATM Machine

insert a bank card

`card_in`

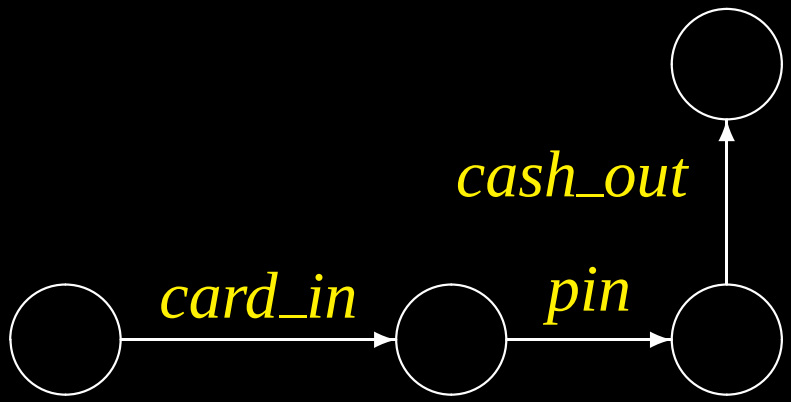
Example: ATM Machine



enter the right pin for that card

```
card_in -> pin
```

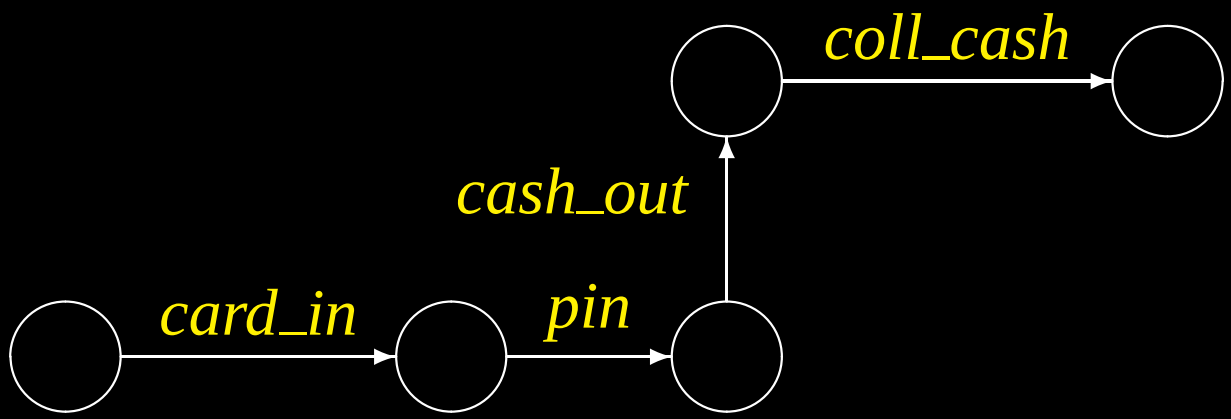
Example: ATM Machine



delivers the cash to the user

```
card_in -> pin -> cash_out
```

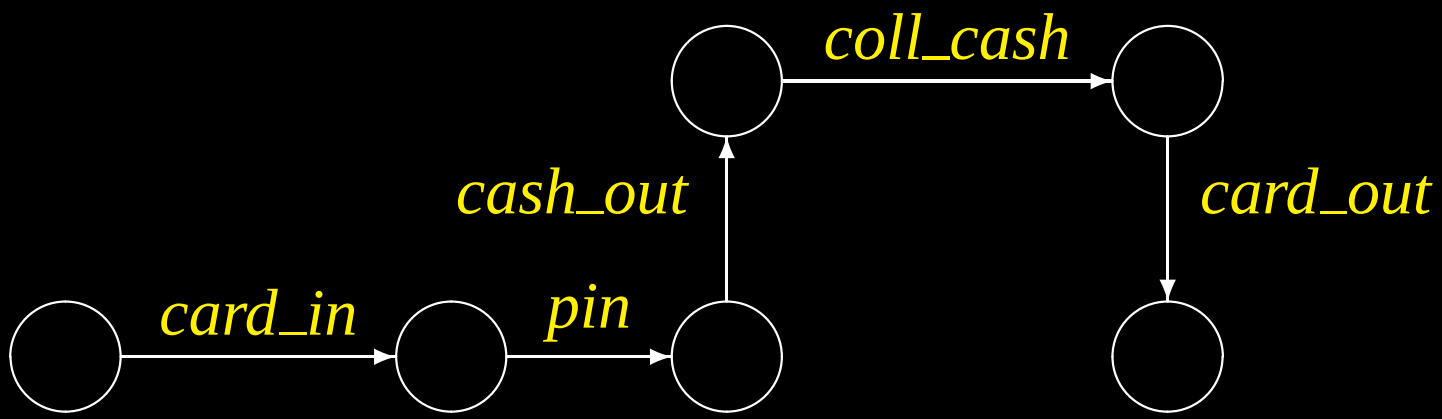
Example: ATM Machine



waits that the user has collected cash

```
card_in -> pin -> cash_out
    -> coll_cash
```

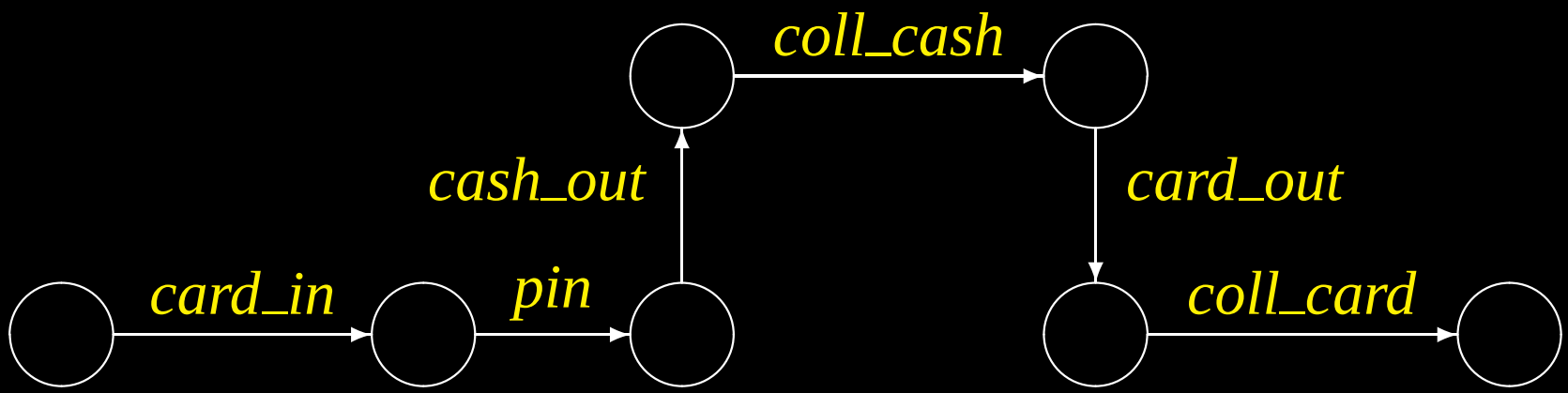
Example: ATM Machine



returns the bank card to the user

```
card_in -> pin -> cash_out
    -> coll_cash -> card_out
```

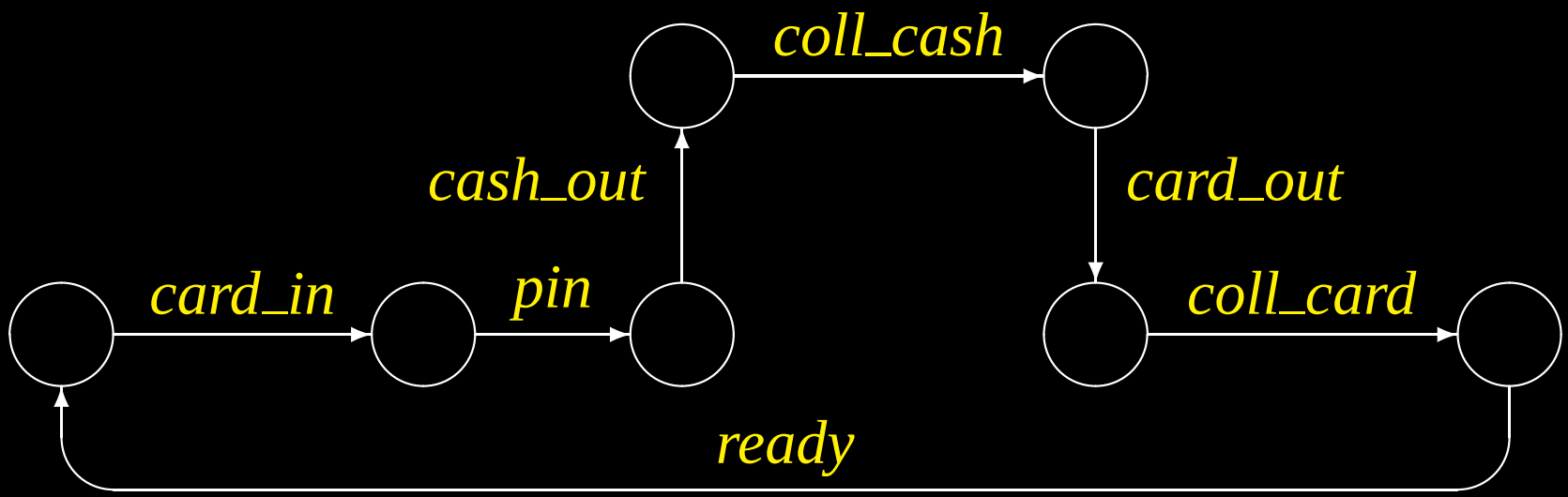
Example: ATM Machine



waits that the user has collected card

```
card_in -> pin -> cash_out
    -> coll_cash -> card_out -> coll_card
```

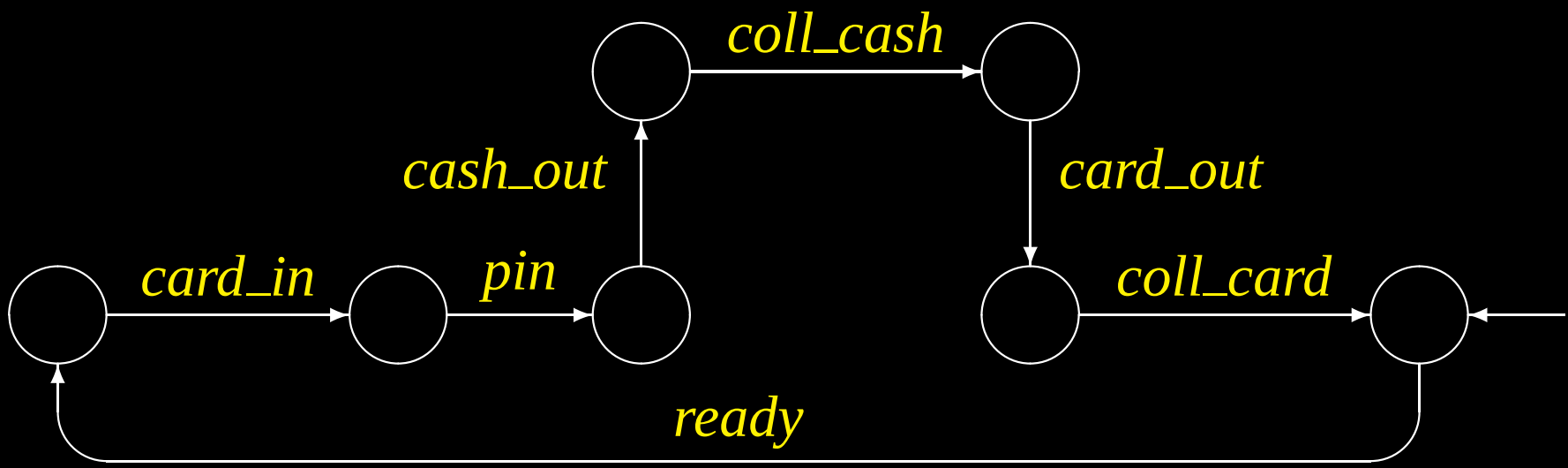
Example: ATM Machine



be ready for a new transaction

```
Machine = card_in -> pin -> cash_out
    -> coll_cash -> card_out -> coll_card
    -> ready -> Machine
```

Example: ATM Machine




```
Machine = ready -> card_in -> pin
    -> cash_out -> coll_cash -> card_out
    -> coll_card -> Machine
```

ATM: CWB-NC Code

CSP Model

```
proc Machine = ready -> card_in
    -> pin -> cash_out -> CashGiven

proc CashGiven = coll_cash
    -> card_out -> CardReturned

proc CardReturned = coll_card -> Machine
```

Simulation with CWB-NC

```
...
cwn-nc>
```

Simulation with CWB-NC

```
...
cwn-nc> load atm-machine.csp
Execution time (user,system,gc,real):(0
cwn-nc>
```

Simulation with CWB-NC

```
...
cwn-nc> load atm-machine.csp
Execution time (user,system,gc,real):(0
cwn-nc> sim Machine
Machine
1:  -- ready --> card_in->cash_out->Cash
cwb-nc-sim>
```

Simulation with CWB-NC

```
...
cwn-nc> load atm-machine.csp
Execution time (user,system,gc,real):(0
cwn-nc> sim Machine
Machine
1:  -- ready --> card_in->cash_out->Cash
cwb-nc-sim>1
card_in->cash_out->CashGiven
1:  -- card_in --> cash_out->CashGiven
cwb-nc-sim>
```

Simulation with CWB-NC

```
...
cwn-nc> load atm-machine.csp
Execution time (user,system,gc,real):(0
cwn-nc> sim Machine
Machine
1:  -- ready --> card_in->cash_out->Cash
cwb-nc-sim>1
card_in->cash_out->CashGiven
1:  -- card_in --> cash_out->CashGiven
cwb-nc-sim>quit
        Execution time (user,system,gc,r
cwb-nc>
```

Simulation with CWB-NC

```
...
cwn-nc> load atm-machine.csp
Execution time (user,system,gc,real):(0
cwn-nc> sim Machine
Machine
1:  -- ready --> card_in->cash_out->Cash
cwb-nc-sim>1
card_in->cash_out->CashGiven
1:  -- card_in --> cash_out->CashGiven
cwb-nc-sim>quit
        Execution time (user,system,gc,r
cwb-nc> quit
shell>
```

ATM: User's Perspective

- Task Knowledge
  - need to have a bank card and to know the pin
  - bank card must be inserted before entering the pin
  - bank card and cash must be collected as soon as they come out

ATM: User's Perspective

- Task Knowledge
  - need to have a bank card and to know the pin
  - bank card must be inserted before entering the pin
  - bank card and cash must be collected as soon as they come out
- Possible User Expectations
  - cash delivered before bank card returned

ATM: User's Perspective

- Task Knowledge
  - need to have a bank card and to know the pin
  - bank card must be inserted before entering the pin
  - bank card and cash must be collected as soon as they come out
- Possible User Expectations
  - cash delivered before bank card returned
  - cash not delivered until bank card collected

ATM: User's Perspective

- Task Knowledge
  - need to have a bank card and to know the pin
  - bank card must be inserted before entering the pin
  - bank card and cash must be collected as soon as they come out
- Possible User Expectations
  - cash delivered before bank card returned
  - cash not delivered until bank card collected
  - bank card returned before cash delivered
  - bank card not returned until cash collected

Task Knowledge

- actions needed to perform it
- preconditions of such actions
- possibly structure of the set of actions (if any)

Task Knowledge

- actions needed to perform it
- preconditions of such actions
- possibly structure of the set of actions (if any)

independently of a specific computer/machine/interface

User Expectations

May depend on

- expertise acquired through use/training

User Expectations

May depend on

- expertise acquired through use/training
  $\Longrightarrow$ user can consciously react to an expectation failure

User Expectations

May depend on

- expertise acquired through use/training
  $\Longrightarrow$ user can consciously react to an expectation failure
- mental model of the way the machine works

User Expectations

May depend on

- expertise acquired through use/training
  $\Longrightarrow$ user can consciously react to an expectation failure
- mental model of the way the machine works
  $\Longrightarrow$ user may not be able to consciously react to an expectation failure

User Expectations

May depend on

- expertise acquired through use/training
  $\Longrightarrow$ user can consciously react to an expectation failure
- mental model of the way the machine works
  $\Longrightarrow$ user may not be able to consciously react to an expectation failure

depend on specific computers/machines/interfaces normally used by the user

ATM: User's Task Knowledge

- need to have a bank card and to know the pin
  not modeled
- bank card must be inserted before entering the pin
  `card_in -> pin`
- bank card and cash must be collected as soon as they come out

ATM: User's Task Knowledge

- need to have a bank card and to know the pin
  not modeled
- bank card must be inserted before entering the pin
  `card_in -> pin`
- bank card and cash must be collected as soon as they come out
  choice between first collecting
  - card
  - cash

CSP Ext Choice Operators

The External Choice (in CWB-NC: `[]`) defines a choice between two possible ordering of actions, whereby the choice is driven by an external process.

CSP Ext Choice Operators

The External Choice (in CWB-NC: `[]`) defines a choice between two possible ordering of actions, whereby the choice is driven by an external process.

Example:

```
(coll_cash -> coll_card -> ...)  []
  (coll_card -> coll_cash -> ...)
```

Choice between action `coll_cash` and action `coll_card`

CSP Ext Choice Operators

The External Choice (in CWB-NC: `[]`) defines a choice between two possible ordering of actions, whereby the choice is driven by an external process.

Example:

```
(coll_cash -> coll_card -> ...)  []
  (coll_card -> coll_cash -> ...)
```

Choice between action `coll_cash` and action `coll_card`

External because driven by the machine

ATM: User Model

```
proc User = card_in -> pin
      -> (CashFirst [] CardFirst)

proc CashFirst = coll_cash
      -> coll_card -> User

proc CardFirst = coll_card
      -> coll_cash -> User
```

ATM: User Model

```
proc User = card_in -> pin
       -> (CashFirst [] CardFirst)

proc CashFirst = coll_cash
      -> coll_card -> User

proc CardFirst = coll_card
      -> coll_cash -> User
```

need a mechanism to compose two processes together in order to allow one process to drive the choice in the other process

CSP Parallel Operators

The Parallel Operator (in CSP: $\|$ and in CWB-NC: $[\,|\ SS\ |\,]$, with SS the set of actions offered by both processes) forces the synchronisation of those actions that are offered by both processes (synchronisation set) while allowing all other actions to occur independently.

CSP Parallel Operators

The Parallel Operator (in CSP: $\|$ and in CWB-NC: `[| SS |]`, with *SS* the set of actions offered by both processes) forces the synchronisation of those actions that are offered by both processes (synchronisation set) while allowing all other actions to occur independently.

Example:

```
Machine || User
```

CSP Parallel Operators

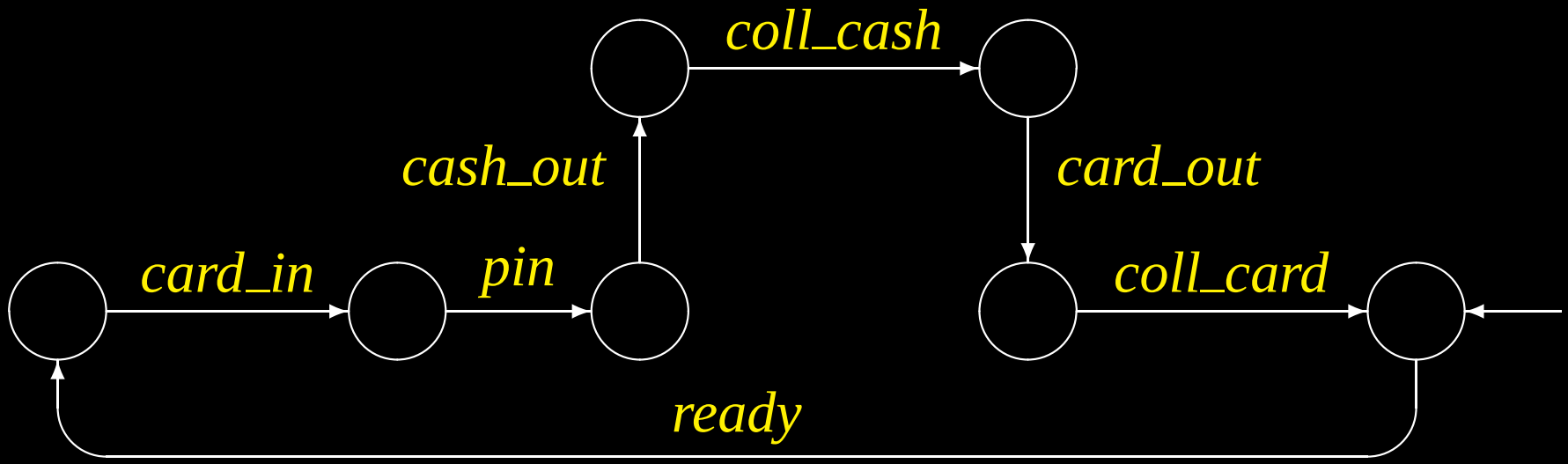
The Parallel Operator (in CSP: $\|$ and in CWB-NC: `[| SS |]`, with *SS* the set of actions offered by both processes) forces the synchronisation of those actions that are offered by both processes (synchronisation set) while allowing all other actions to occur independently.

Example:

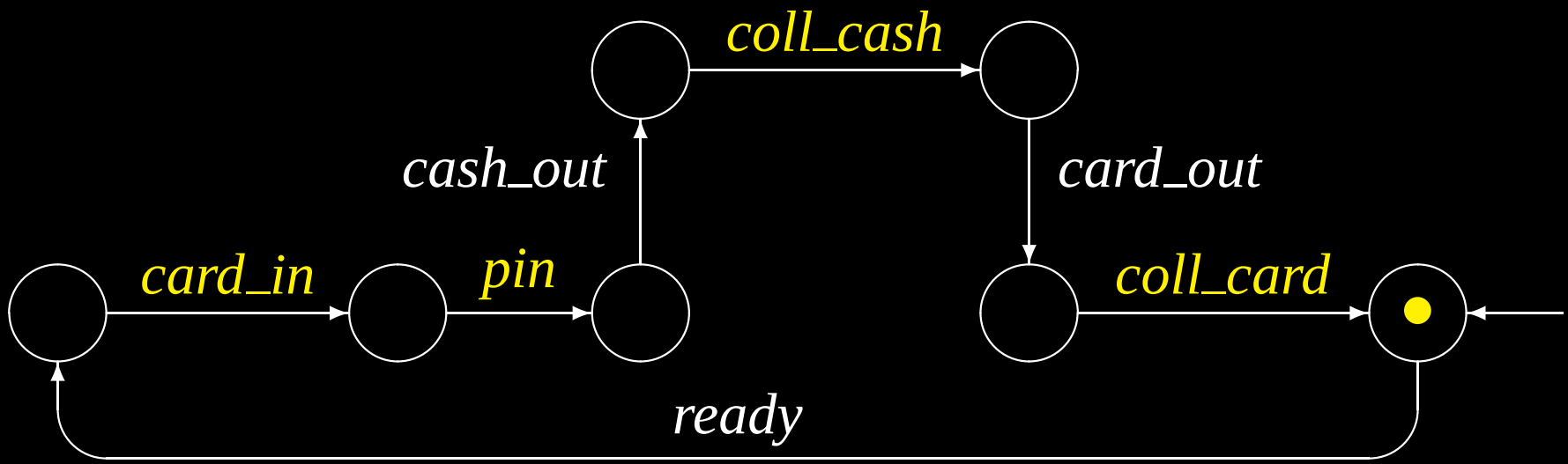
```
Machine || User

Machine [| {card_in,pin,coll_cash,
            coll_card} |] User
```

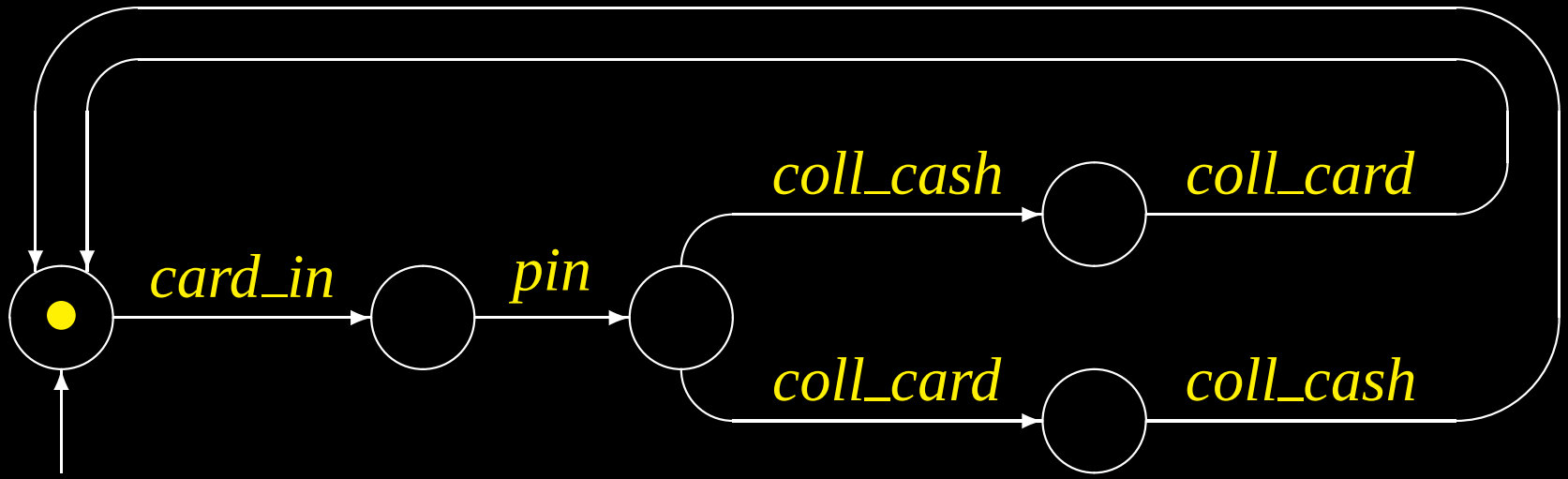
Synchronisation between process `Machine` and process `User` on the set of common actions `{card_in,pin,coll_cash,coll_card}`

ATM: Machine and User

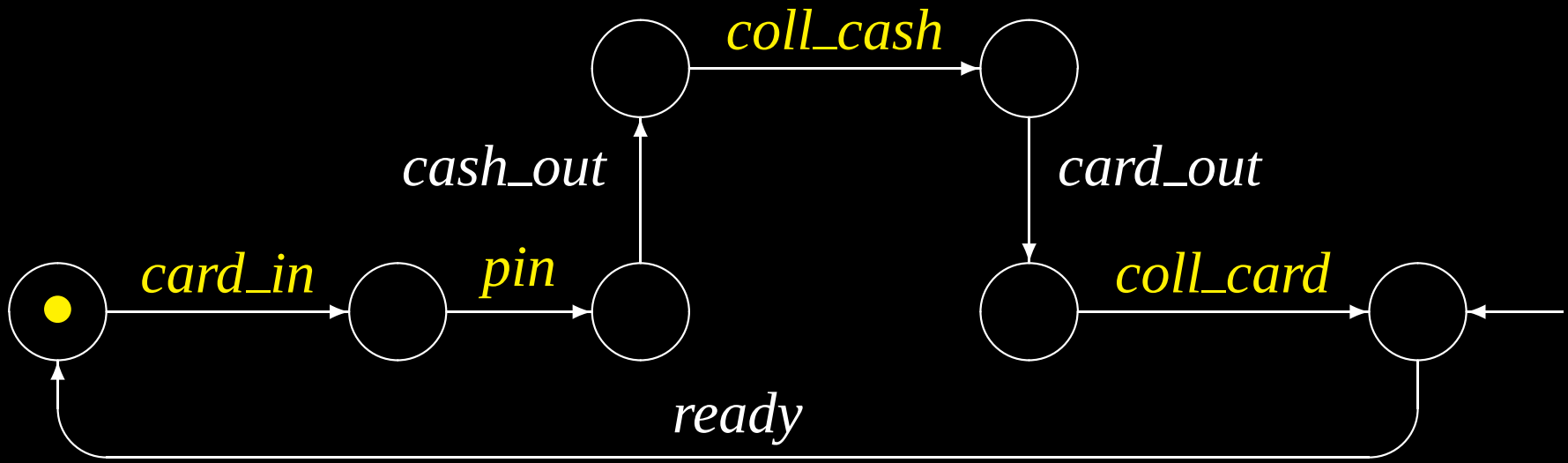


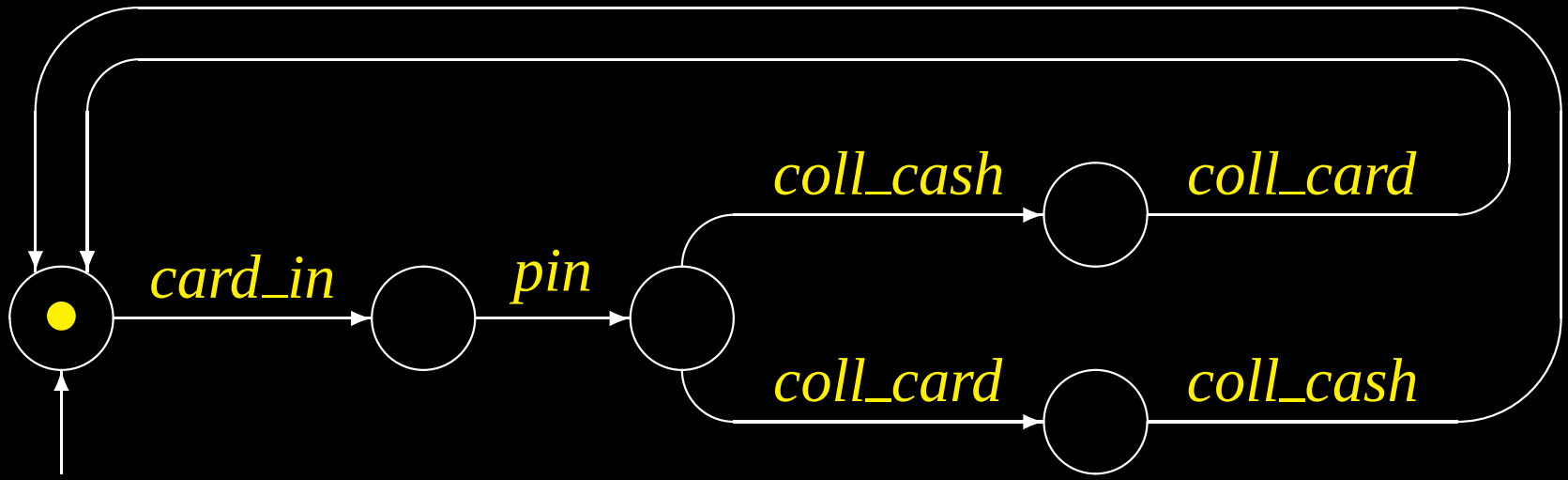
ATM: Machine and User



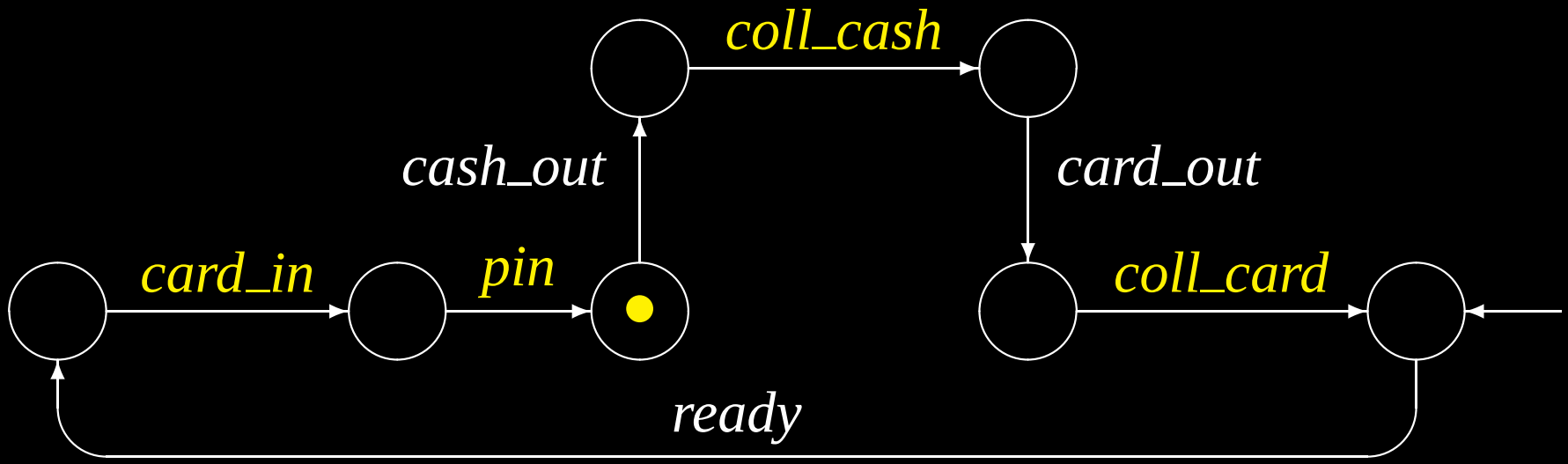


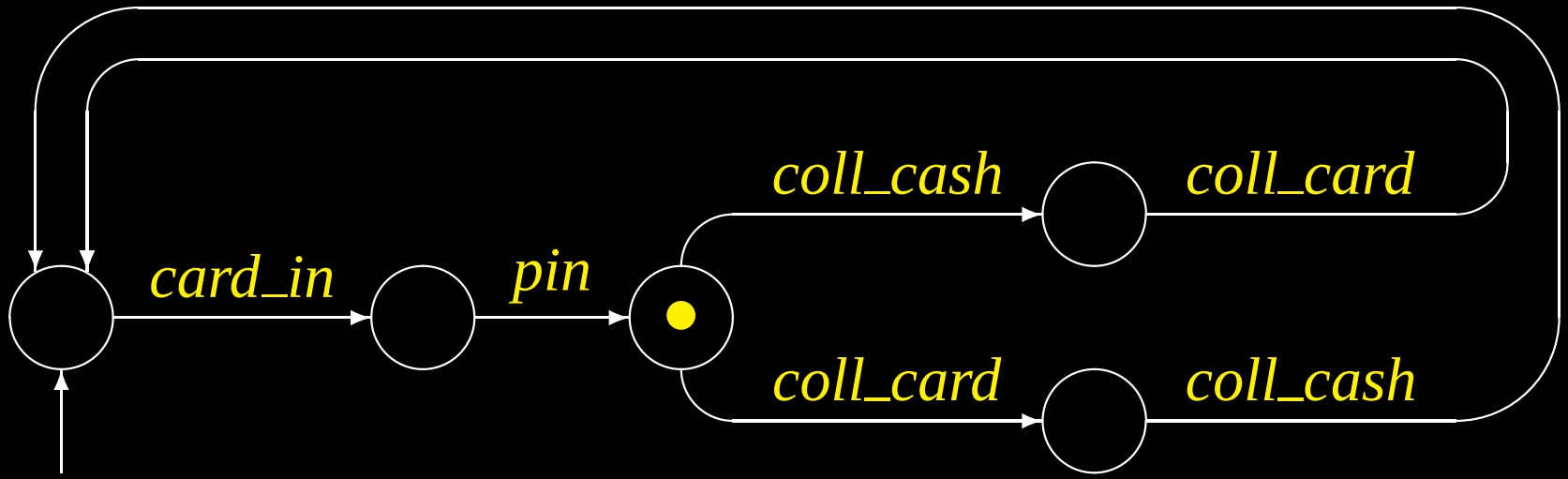
ATM: Machine and User



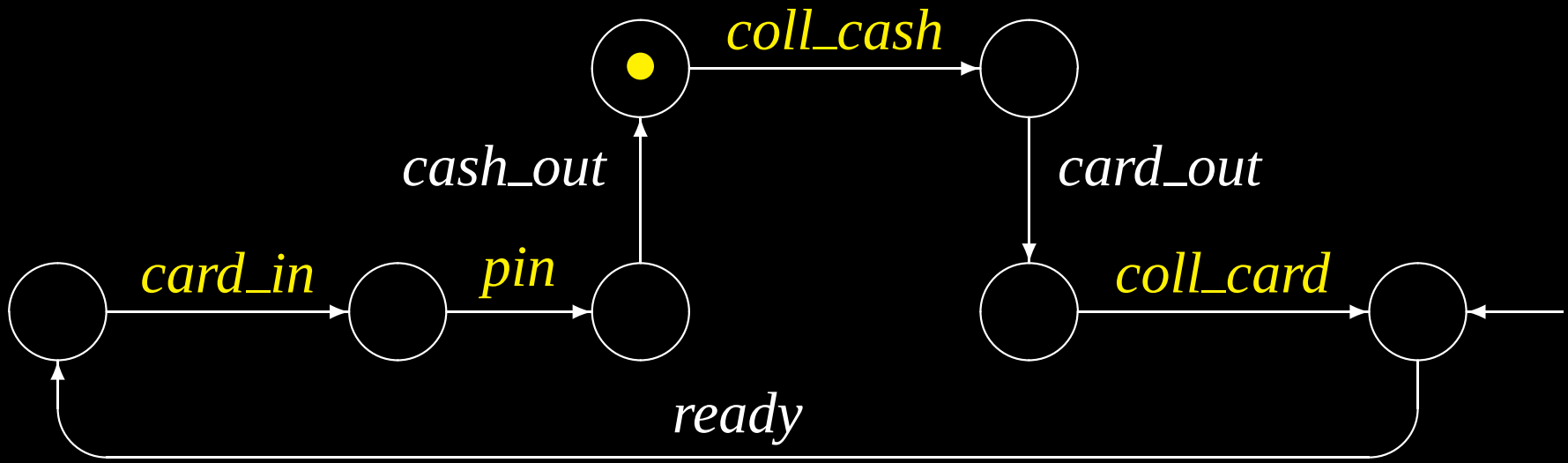


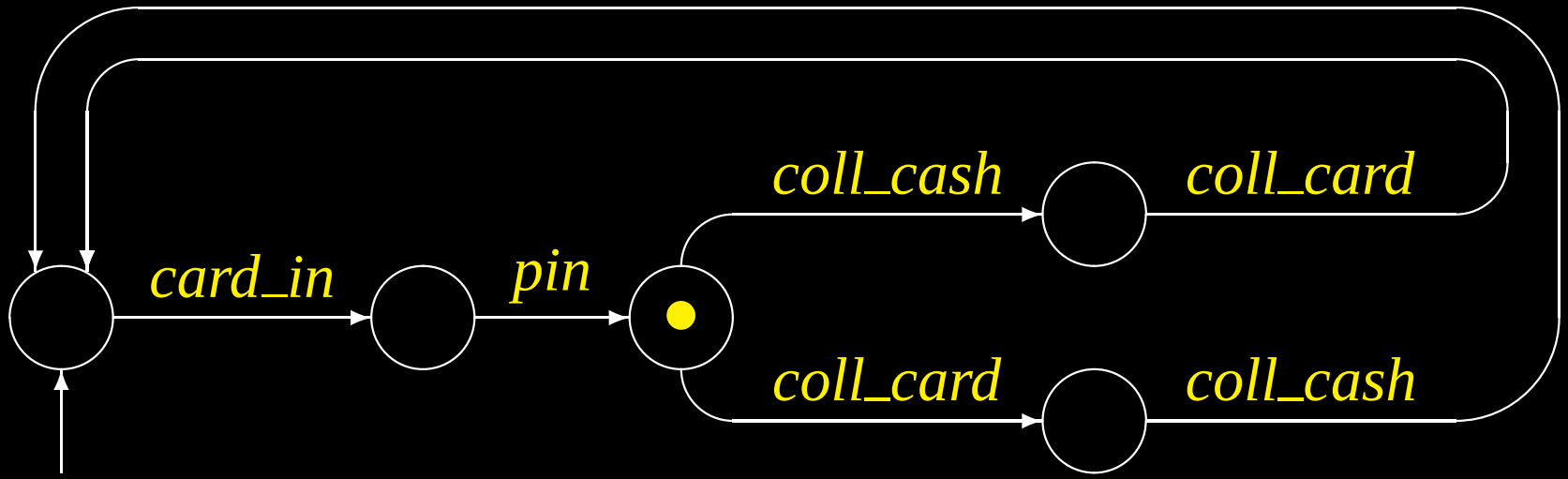
ATM: Machine and User



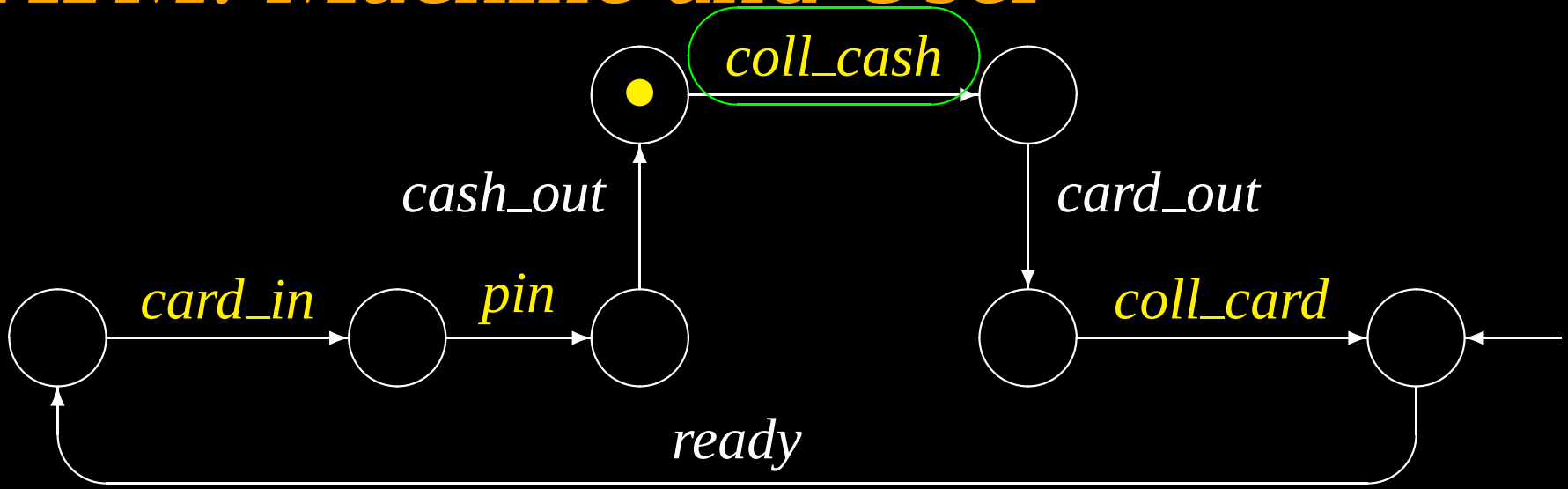


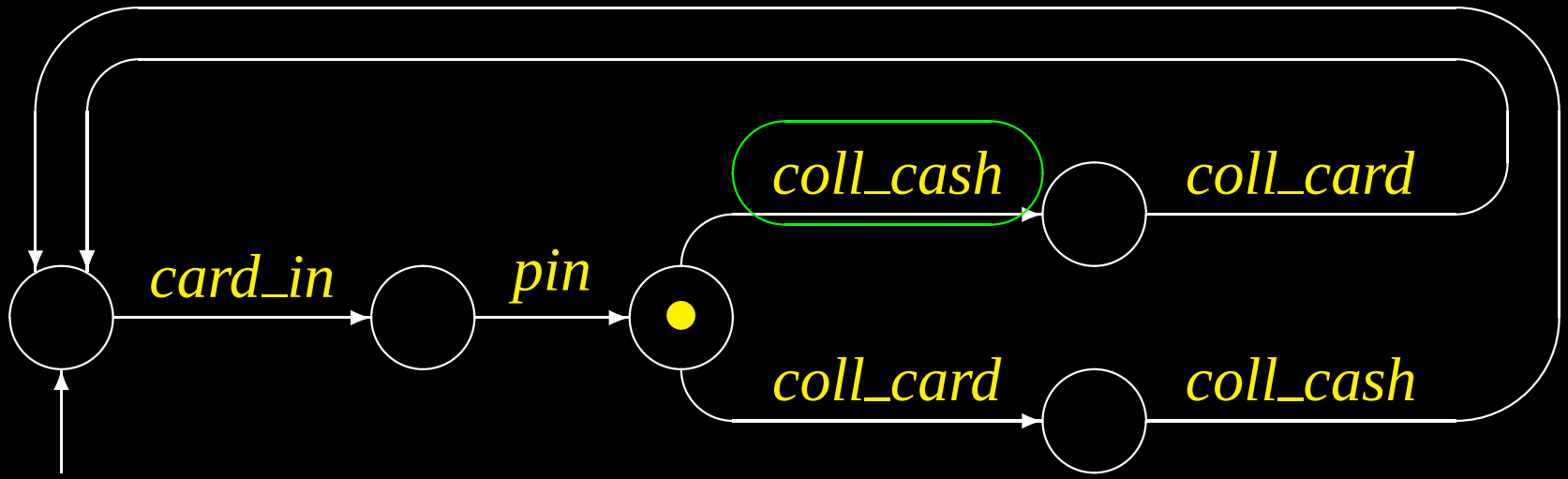
ATM: Machine and User



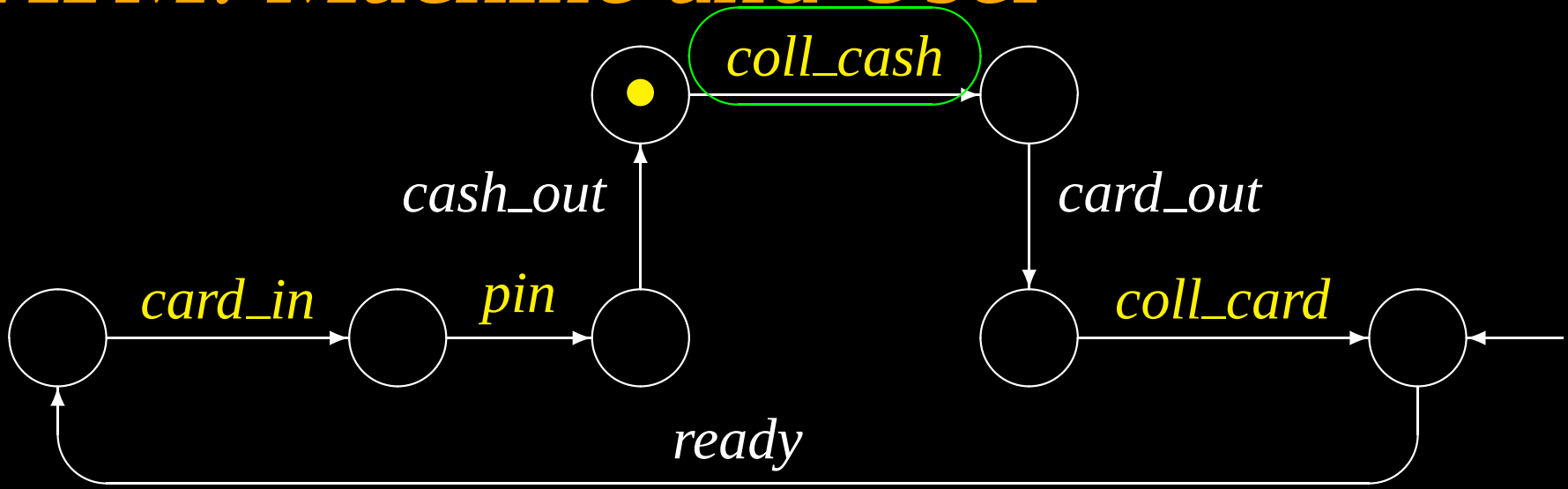


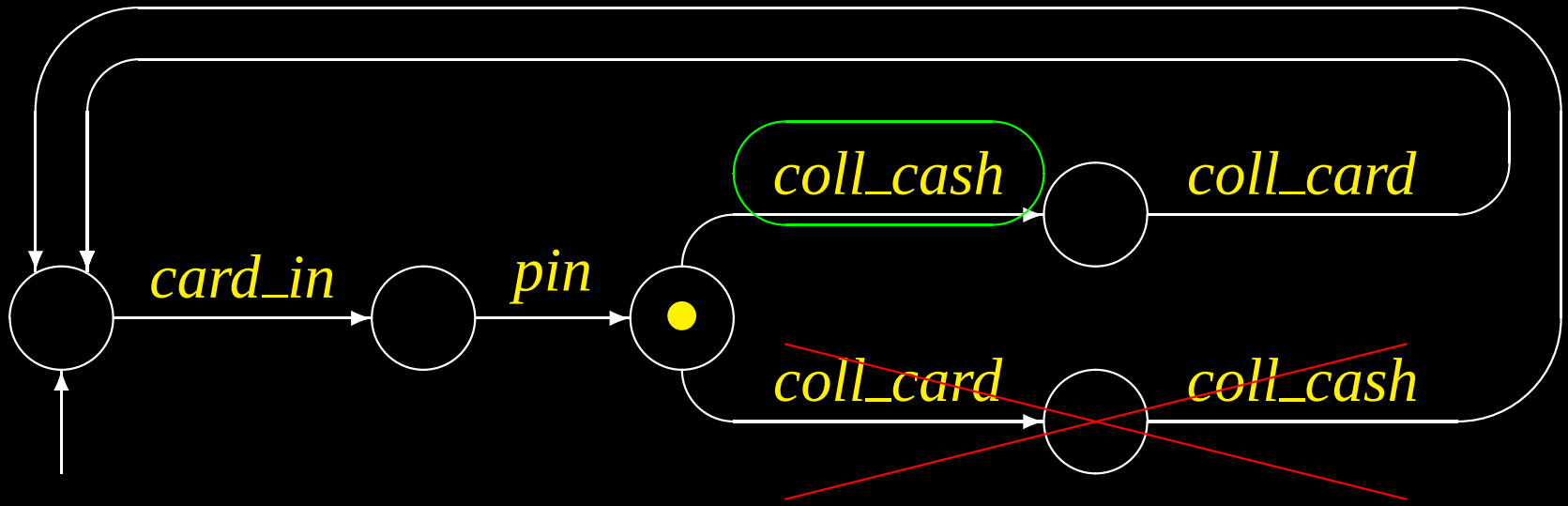
ATM: Machine and User



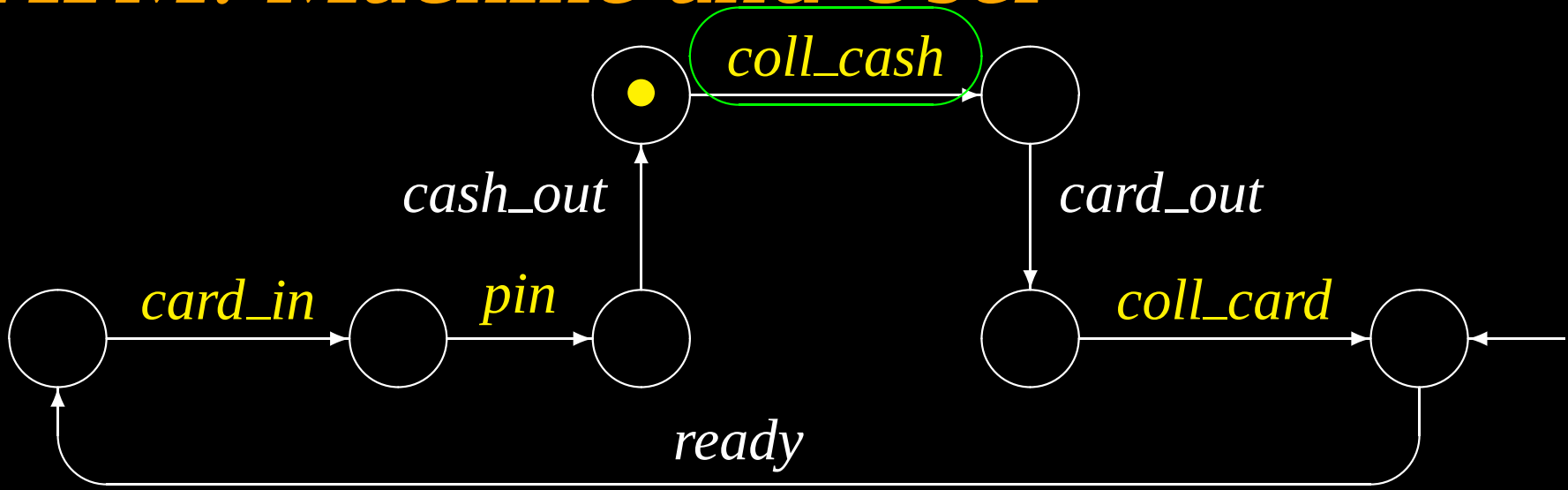


ATM: Machine and User

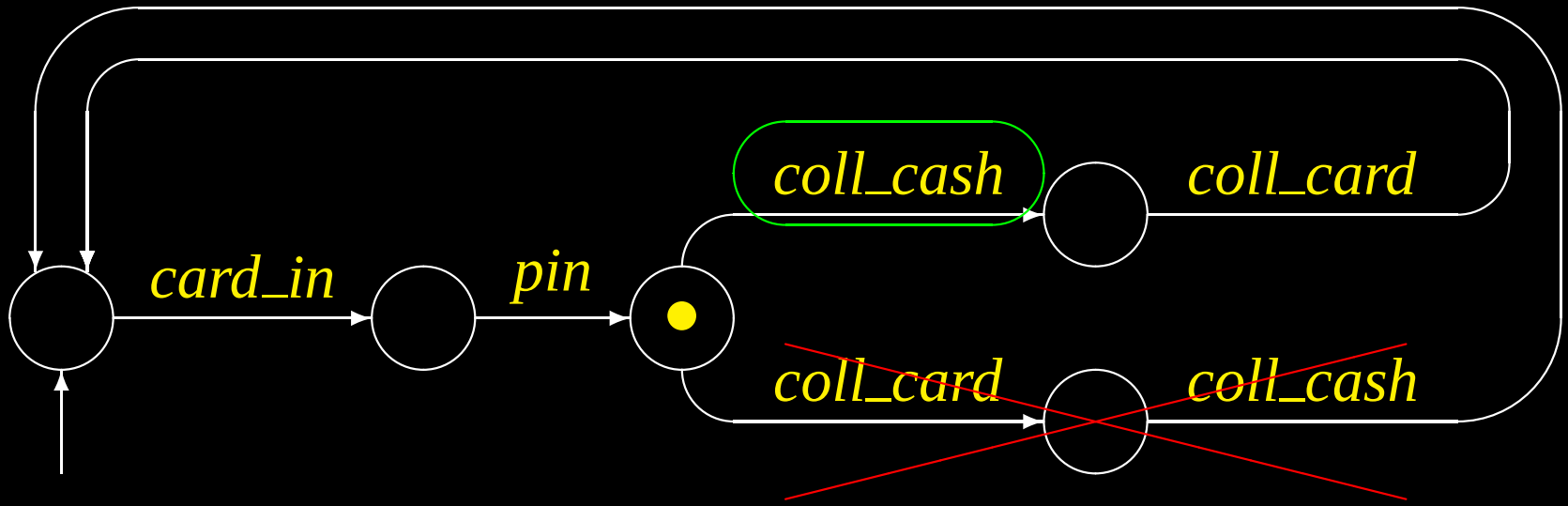




ATM: Machine and User



Process *Machine* constrains process *User*



Constraining Processes

- Process *Machine* constrains process *User* to take alternative

```
coll_cash -> coll_card
```

Constraining Processes

- Process *Machine* constrains process *User* to take alternative

```
coll_cash -> coll_card
```

- collection order is driven by the machine

Constraining Processes

- Process *Machine* constrains process *User* to take alternative

```
coll_cash -> coll_card
```

- collection order is driven by the machine
- can you suggest any action offered by the machine and driven by the user?

Constraining Processes

- Process *Machine* constrains process *User* to take alternative

```
coll_cash -> coll_card
```

- collection order is driven by the machine
- can you suggest any action offered by the machine and driven by the user?
- transaction selection is offered by the machine but is driven by the user

Constraining Processes

- Process *Machine* constrains process *User* to take alternative

```
coll_cash -> coll_card
```

- collection order is driven by the machine
- can you suggest any action offered by the machine and driven by the user?
- transaction selection is offered by the machine but is driven by the user
- how would you model this?

ATM: Transaction Sel — M

```
proc Machine = ready -> card_in
    -> pin -> Selection

Selection = select_cash
    -> cash_out -> CashGiven
      [] select_stat
    -> stat_out -> StatGiven

proc CashGiven = coll_cash
    -> card_out -> CardReturned

proc StatGiven = coll_stat
    -> card_out -> CardReturned

proc CardReturned = coll_card -> Machine
```

ATM: Transaction Sel — U1

```
proc User = card_in -> pin -> select_cash
      -> (CashFirst [] CardFirst)

proc CashFirst = coll_cash
      -> coll_card -> User

proc CardFirst = coll_card
      -> coll_cash -> User
```

ATM: Transaction Sel — U1

```
proc User = card_in -> pin -> select_cash
      -> (CashFirst [] CardFirst)

proc CashFirst = coll_cash
      -> coll_card -> User

proc CardFirst = coll_card
      -> coll_cash -> User
```

What is the synchronisation set?

ATM: Transaction Sel — U1

```
proc User = card_in -> pin -> select_cash
      -> (CashFirst [] CardFirst)

proc CashFirst = coll_cash
      -> coll_card -> User

proc CardFirst = coll_card
      -> coll_cash -> User
```

What is the synchronisation set?

`card_in`

ATM: Transaction Sel — U1

```
proc User = card_in -> pin -> select_cash
      -> (CashFirst [] CardFirst)

proc CashFirst = coll_cash
      -> coll_card -> User

proc CardFirst = coll_card
      -> coll_cash -> User
```

What is the synchronisation set?

`card_in`

`pin`

ATM: Transaction Sel — U1

```
proc User = card_in -> pin -> select_cash
      -> (CashFirst [] CardFirst)

proc CashFirst = coll_cash
      -> coll_card -> User

proc CardFirst = coll_card
      -> coll_cash -> User
```

What is the synchronisation set?

```
card_in
pin
select_cash
```

ATM: Transaction Sel — U1

```
proc User = card_in -> pin -> select_cash
      -> (CashFirst [] CardFirst)

proc CashFirst = coll_cash
      -> coll_card -> User

proc CardFirst = coll_card
      -> coll_cash -> User
```

What is the synchronisation set?

```
card_in
pin
select_cash
coll_cash
```

ATM: Transaction Sel — U1

```
proc User = card_in -> pin -> select_cash
      -> (CashFirst [] CardFirst)

proc CashFirst = coll_cash
      -> coll_card -> User

proc CardFirst = coll_card
      -> coll_cash -> User
```

What is the synchronisation set?

card_in
pin
select_cash
coll_cash
coll_card

ATM: Transaction Sel — U1

```
proc User = card_in -> pin -> select_cash
      -> (CashFirst [] CardFirst)

proc CashFirst = coll_cash
      -> coll_card -> User

proc CardFirst = coll_card
      -> coll_cash -> User
```

What is the synchronisation set?

card_in
pin
select_cash
coll_cash
coll_card
select_stat

ATM: Deadlock

System trace (= Machine trace):

```
ready card_in pin select_stat
stat_out coll_stat card_out
```

The agent has no transitions

ATM: Deadlock

System trace (= Machine trace):

```
ready card_in pin select_stat
stat_out coll_stat card_out
```

The agent has no transitions

User trace:

```
card_in pin
```

ATM: Deadlock

System trace (= Machine trace):

```
ready card_in pin select_stat
stat_out coll_stat card_out
```

The agent has no transitions

User trace:

```
card_in pin
```

The user may only perform `select_cash`

The machine may only perform `coll_card`

ATM: Deadlock

System trace (= Machine trace):

`ready card_in pin select_stat stat_out coll_stat card_out`

The agent has no transitions

User trace:

`card_in pin`

The user may only perform `select_cash`

The machine may only perform `coll_card`

Both actions are in the synchronisation set
$\Longrightarrow$ deadlock!

ATM: User's Selection

- Previously user always select cash
- What if the user may perform either `select_cash` or `select_stat`?

ATM: User's Selection

- Previously user always select cash
- What if the user may perform either `select_cash` or `select_stat`?
- The selection is made by the user. It is not made by the machine!
- How to model this asymmetry?

CSP Int Choice Operators

The Internal Choice (in CWB-NC: $|\sim|$) defines an intenal choice between two possible ordering of actions.

CSP Int Choice Operators

The Internal Choice (in CWB-NC: `|~|`) defines an intenal choice between two possible ordering of actions.

Example:

```
(select_cash -> ...)
|~| (select_stat -> ...)
```

User's choice between action `select_cash` and action `select_stat` which are both offered by the machine

CSP Int Choice Operators

The Internal Choice (in CWB-NC: `|~|`) defines an intenal choice between two possible ordering of actions.

Example:

```
(select_cash -> ...)
|~| (select_stat -> ...)
```

User's choice between action `select_cash` and action `select_stat` which are both offered by the machine

Internal because made by the user

ATM: Transaction Sel — U2

```
proc User = card_in -> pin ->
    ( select_cash -> Withdrawal
   |~| select_stat -> Statement )

proc Withdrawal =
      coll_cash -> coll_card -> User
    [] coll_card -> coll_cash -> User

proc Statement =
     coll_stat -> coll_card -> User
   [] coll_card -> coll_stat -> User
```

ATM: Transaction Sel — U2

```
proc User = card_in -> pin ->
    ( select_cash -> Withdrawal
   |~| select_stat -> Statement )

proc Withdrawal =
      coll_cash -> coll_card -> User
    [] coll_card -> coll_cash -> User

proc Statement =
     coll_stat -> coll_card -> User
   [] coll_card -> coll_stat -> User

proc System = Machine [|{card_in,pin,
    select_cash,select_stat,coll_cash,
    coll_stat,coll_card} |] User
```

HCI Concepts

HCI and Interactive Systems

Humans (Users) interact with Computers

- to achieve goals
- by performing tasks

HCI and Interactive Systems

Humans (Users) interact with Computers

- to achieve goals
- by performing tasks

Inteactive Systems are designed to assist user

User: first priority in the requirements

HCI and Interactive Systems

Humans (Users) interact with Computers

- to achieve goals
- by performing tasks

Inteactive Systems are designed to assist user

User: first priority in the requirements

Need to understand

- capabilities
- limitations

of the user

Relevant Human Aspects

(which have a bearing with Computer Systems)

- how humans perceive the world around them
- how they store information and solve problems
- how they physically manipulate objects

Relevant Human Aspects

(which have a bearing with Computer Systems)

- how humans perceive the world around them
- how they store information and solve problems
- how they physically manipulate objects

$\Longrightarrow$ (simplified) model of human processing

Relevant Human Aspects

(which have a bearing with Computer Systems)

- how humans perceive the world around them
- how they store information and solve problems
- how they physically manipulate objects

$\Longrightarrow$ (simplified) model of human processing based on

- Computer Analogy
- Information Processing Theory

Computer Analogy

Computers take a symbolic input, recode it, make decisions about the recoded input, make new expressions from it, store some or all of the input, and give back a symbolic input.

By analogy that is what most cognitive psychology is about.
It is about how most people take in information, how they recode and remember it, how they make decisions, how they transform their internal knowledge states, and how they translate these states into behavioural outputs.

[Lachman et al. 79]
R. Lachman, J. L. Lachman, E. C. Butterfield.
Cognitive Psychology and Information Processing.
Lawrence Erlbaum, 1979.

Organisational Level Analogy

- Central Processing Unit analogous to the mechanism responsible for mental operations to manipulate information
- Information Store analogous to long-term memory
- Information Buffer analogous to short-term memory

Organisational Level Analogy

- Central Processing Unit analogous to the mechanism responsible for mental operations to manipulate information
- Information Store analogous to long-term memory
- Information Buffer analogous to short-term memory

Unlikely computers humans are also influenced by external factors, such as social and organisational environment.

Information Processing Theory

- Behaviourism: Psychology should be based solely on observable events, with no mentalistic concepts

Information Processing Theory

- Behaviourism: Psychology should be based solely on observable events, with no mentalistic concepts
- Information Processing defines models to characterise the nature of mental processes

Information Processing Theory

- Behaviourism: Psychology should be based solely on observable events, with no mentalistic concepts
- Information Processing defines models to characterise the nature of mental processes
  - based on computer analogy

Information Processing Theory

- Behaviourism: Psychology should be based solely on observable events, with no mentalistic concepts
- Information Processing defines models to characterise the nature of mental processes
  - based on computer analogy
  - use experiments based on
    - analysis of response
    - subjective analysis

    to confirm and extend the theory

Model Human Processor

developed by Card, Moran and Newell in 1983 [Card et al. 83], consists of:

- perceptual system handling sensory stimulus form the outside world
- motor system which control actions
- cognitive system which connects the other two subsystems

Model Human Processor

developed by Card, Moran and Newell in 1983 [Card et al. 83], consists of:

- perceptual system handling sensory stimulus form the outside world
- motor system which control actions
- cognitive system which connects the other two subsystems

each equiped with its own processor and memory (short-term and long-tem).

Model Human Processor

developed by Card, Moran and Newell in 1983 [Card et al. 83], consists of:

- perceptual system handling sensory stimulus form the outside world
- motor system which control actions
- cognitive system which connects the other two subsystems

each equiped with its own processor and memory (short-term and long-tem). In addition

- principles of operation dictates the behaviour of the system under certain conditions

Simplified Generic Model

A human system is an intelligent information processing system

Simplified Generic Model

A human system is an intelligent information processing system consisting of:

- Input-Output: senses and responders (or effectors)
  involves some low-level processing

Simplified Generic Model

A human system is an intelligent information processing system consisting of:

- Input-Output: senses and responders (or effectors)
  involves some low-level processing
- Memory (short-term and long-tem)

Simplified Generic Model

A human system is an intelligent information processing system consisting of:

- Input-Output: senses and responders (or effectors)
  involves some low-level processing
- Memory (short-term and long-tem)
- Processing
  - problem solving
  - learning

Simplified Generic Model

A human system is an intelligent information processing system consisting of:

- Input-Output: senses and responders (or effectors)
  involves some low-level processing
- Memory (short-term and long-tem)
- Processing
  - problem solving
  - learning and consequentely
  - making mistakes

User Knowledge

- goal

User Knowledge

- goal
- about task
  - actions to perform it

User Knowledge

- goal
- about task
  - actions to perform it
  - (possibly) structure of the set of actions

User Knowledge

- goal
- about task
  - actions to perform it
  - (possibly) structure of the set of actions

independently of a specific computer/machine/interface

User Knowledge

- goal
- about task
  - actions to perform it
  - (possibly) structure of the set of actions independently of a specific computer/machine/interface
- about machine
  - expertise acquired through use/training

User Knowledge

- goal
- about task
  - actions to perform it
  - (possibly) structure of the set of actions

  independently of a specific computer/machine/interface
- about machine
  - expertise acquired through use/training
  - mental model

User Actions

- mental processing structured set of actions

User Actions

- mental processing structured set of actions
- interaction
  - driven by the mental model
  - triggered by the machine

User Actions

- mental processing structured set of actions
- interaction
  - driven by the mental model
  - triggered by the machine

involve the use of human memory

Human Memory

Sensory Memories

Short-term Memory
or
Working Memory

Long-term Memory

Human Memory

Sensory Memories

Iconic | Echoic | Haptic

Short-term Memory

or

Working Memory

Long-term Memory

Human Memory

Sensory Memories

Iconic | Echoic | Haptic

information persists for $<$ 500 ms

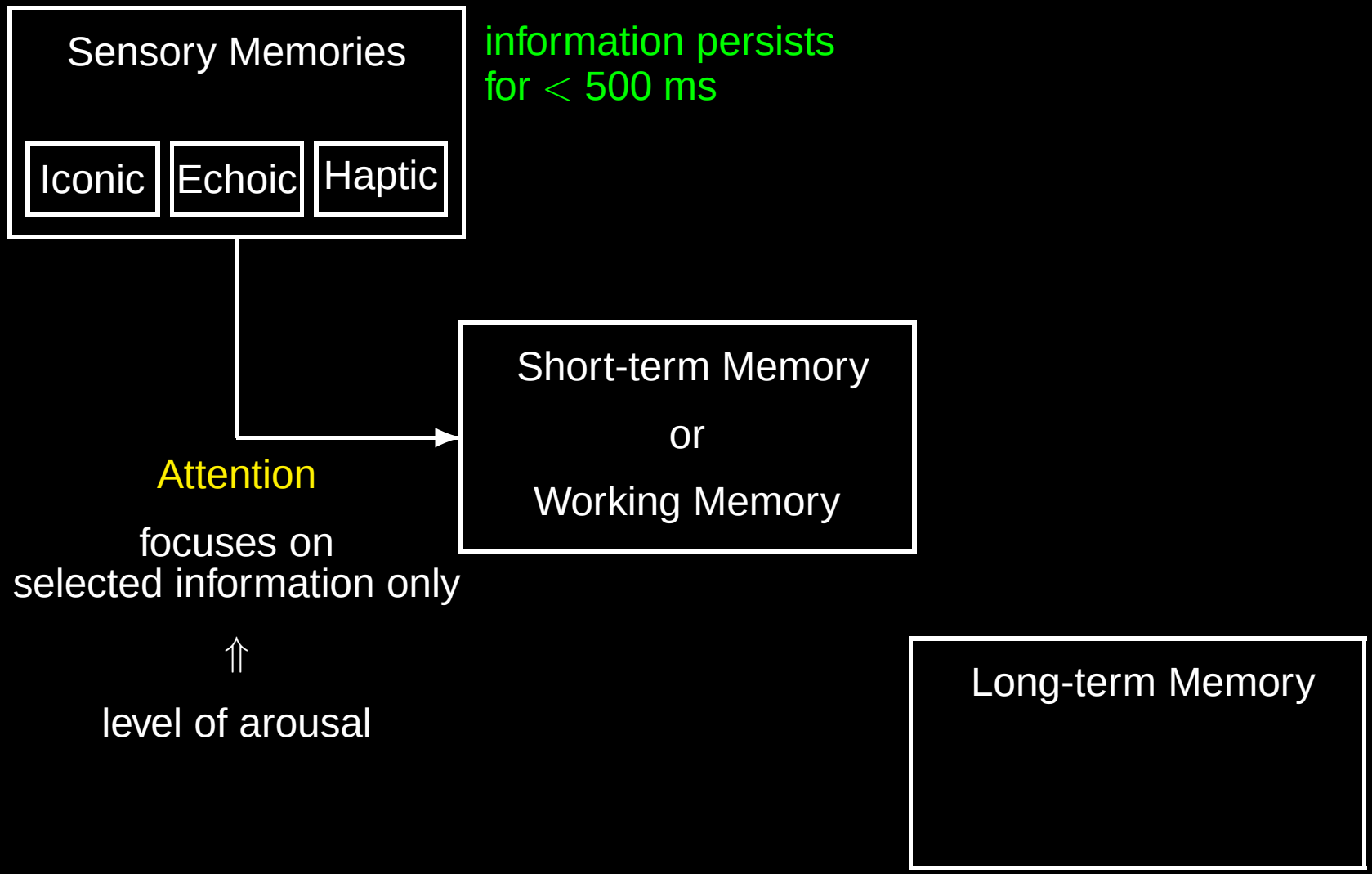
Short-term Memory

or

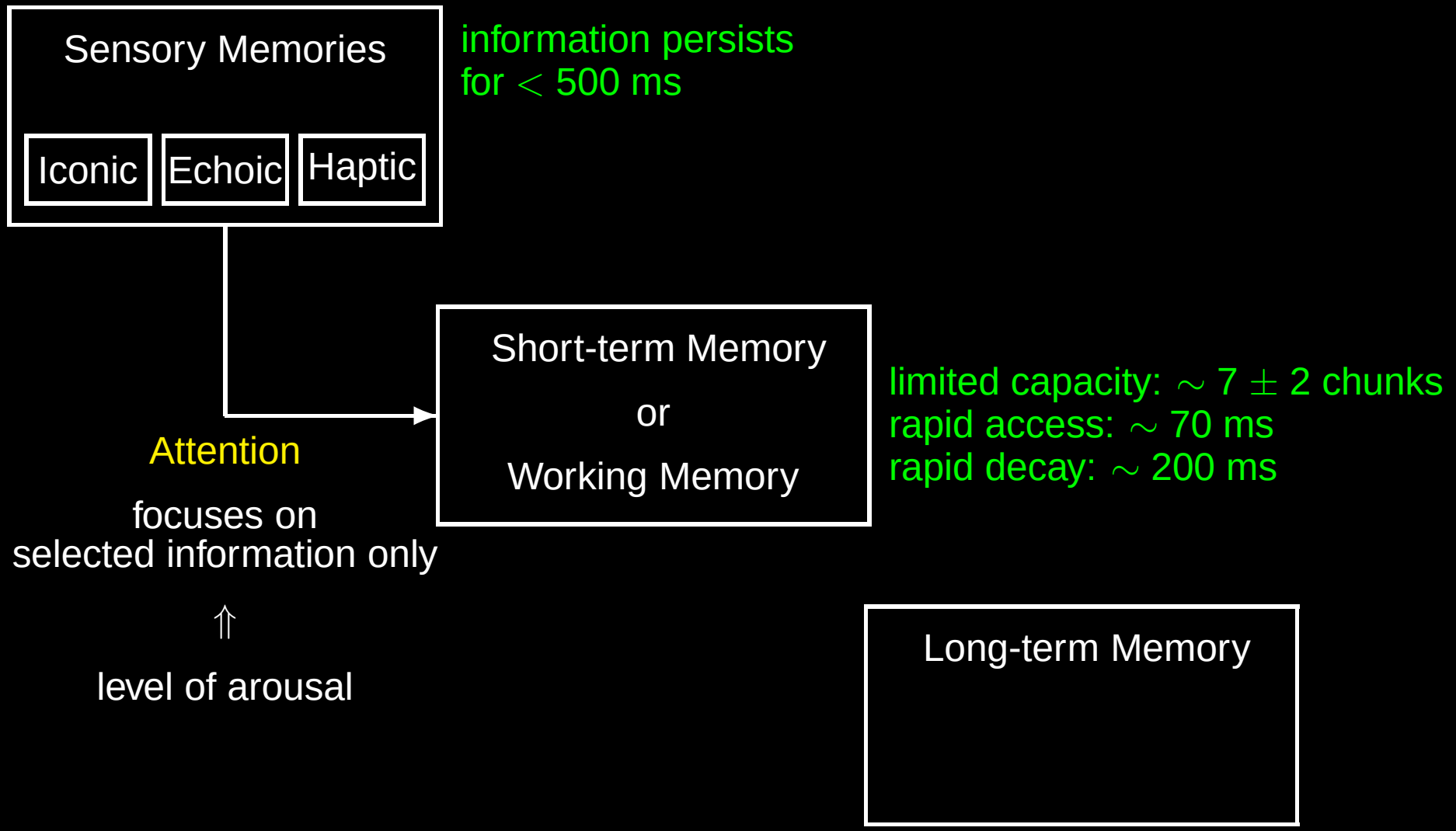
Working Memory

Long-term Memory

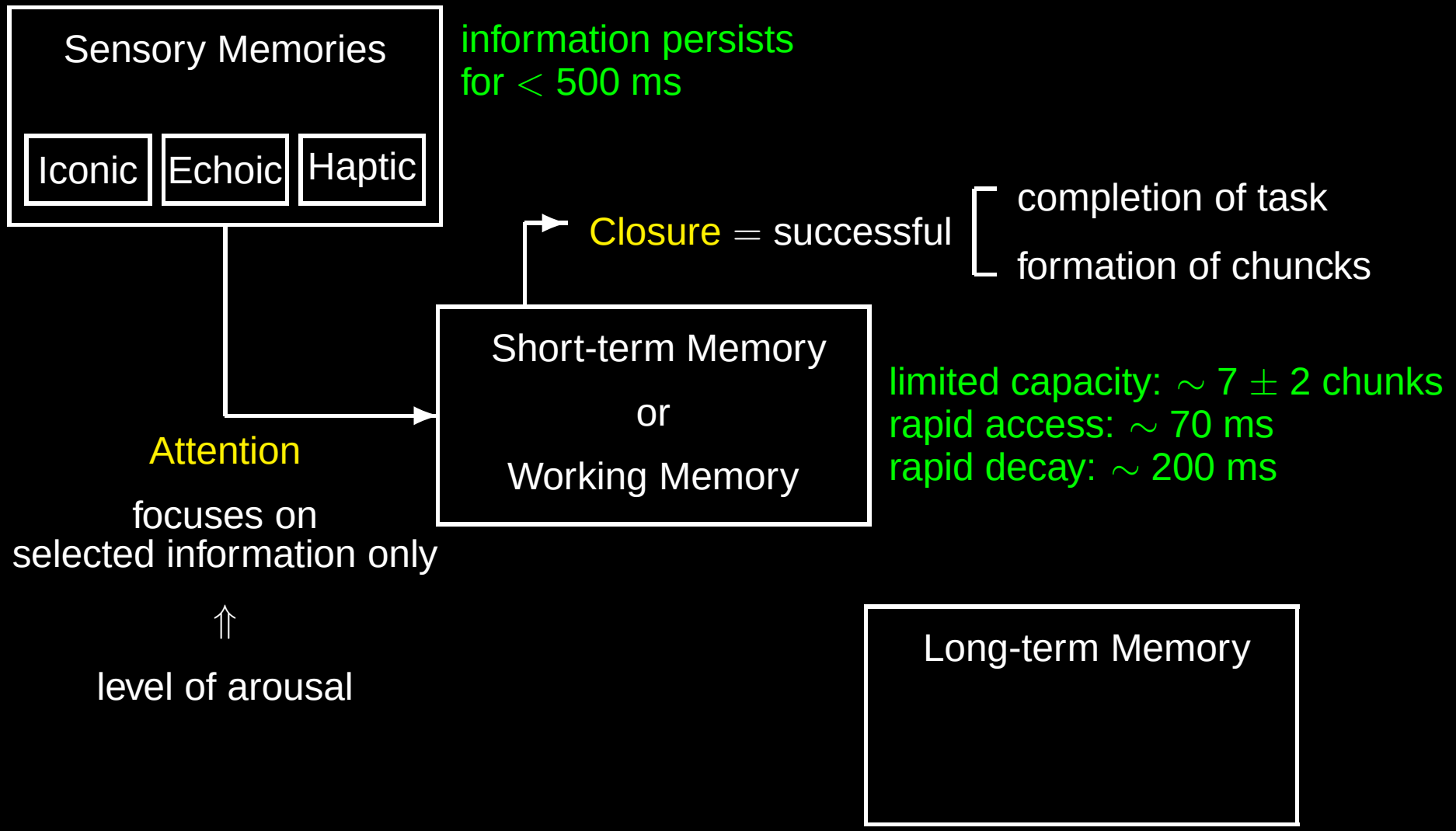
Human Memory



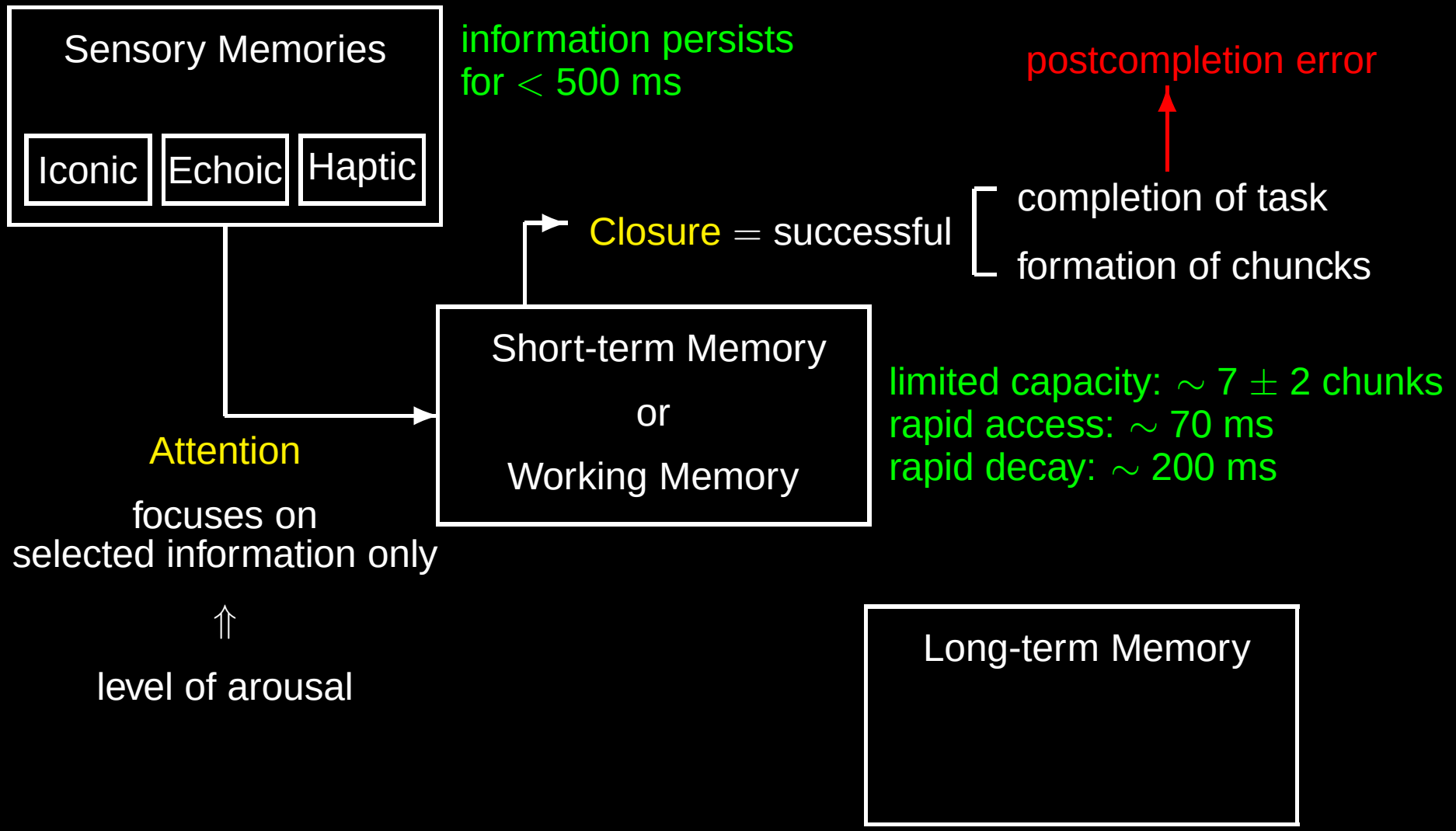
Human Memory



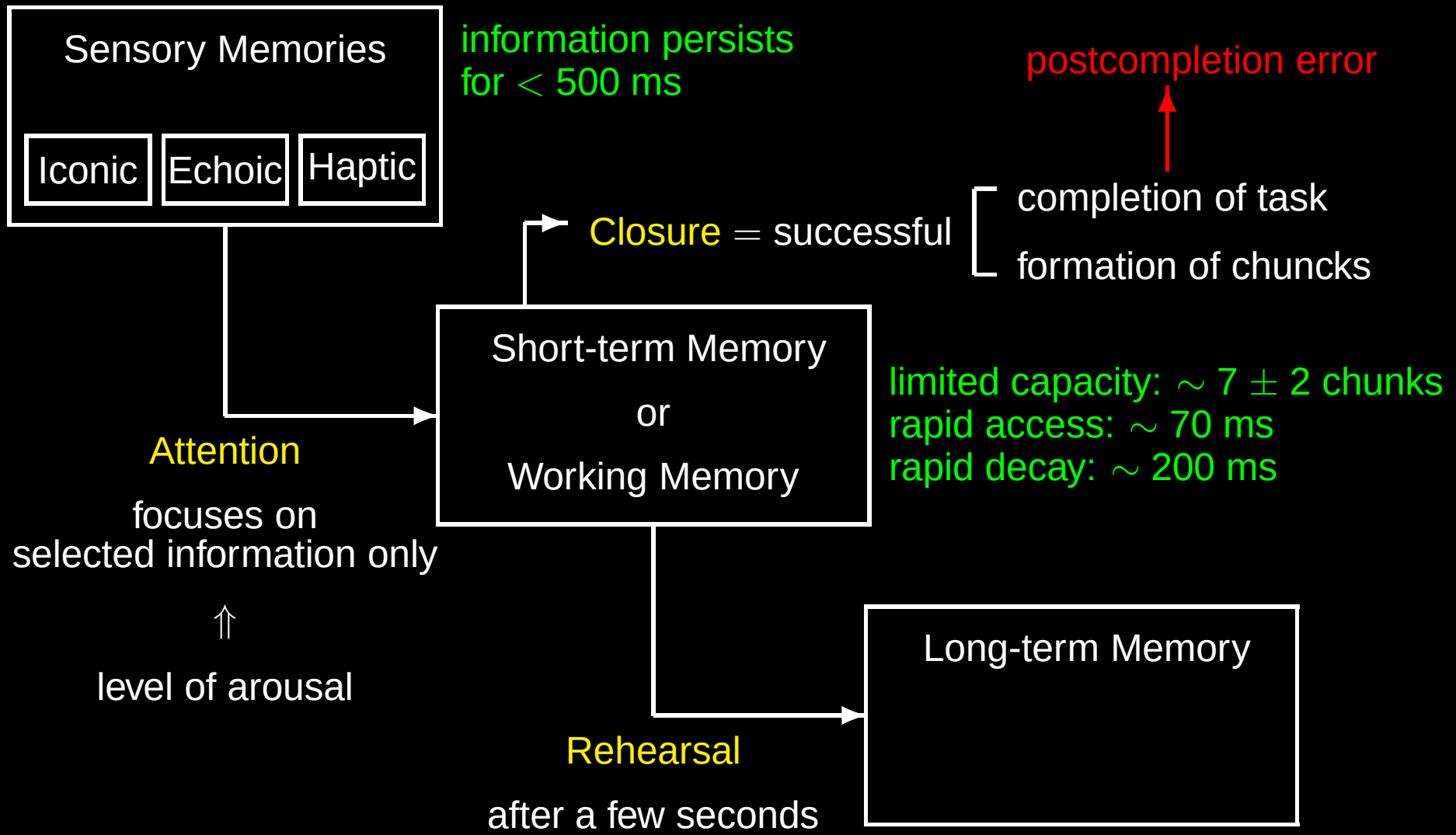
Human Memory



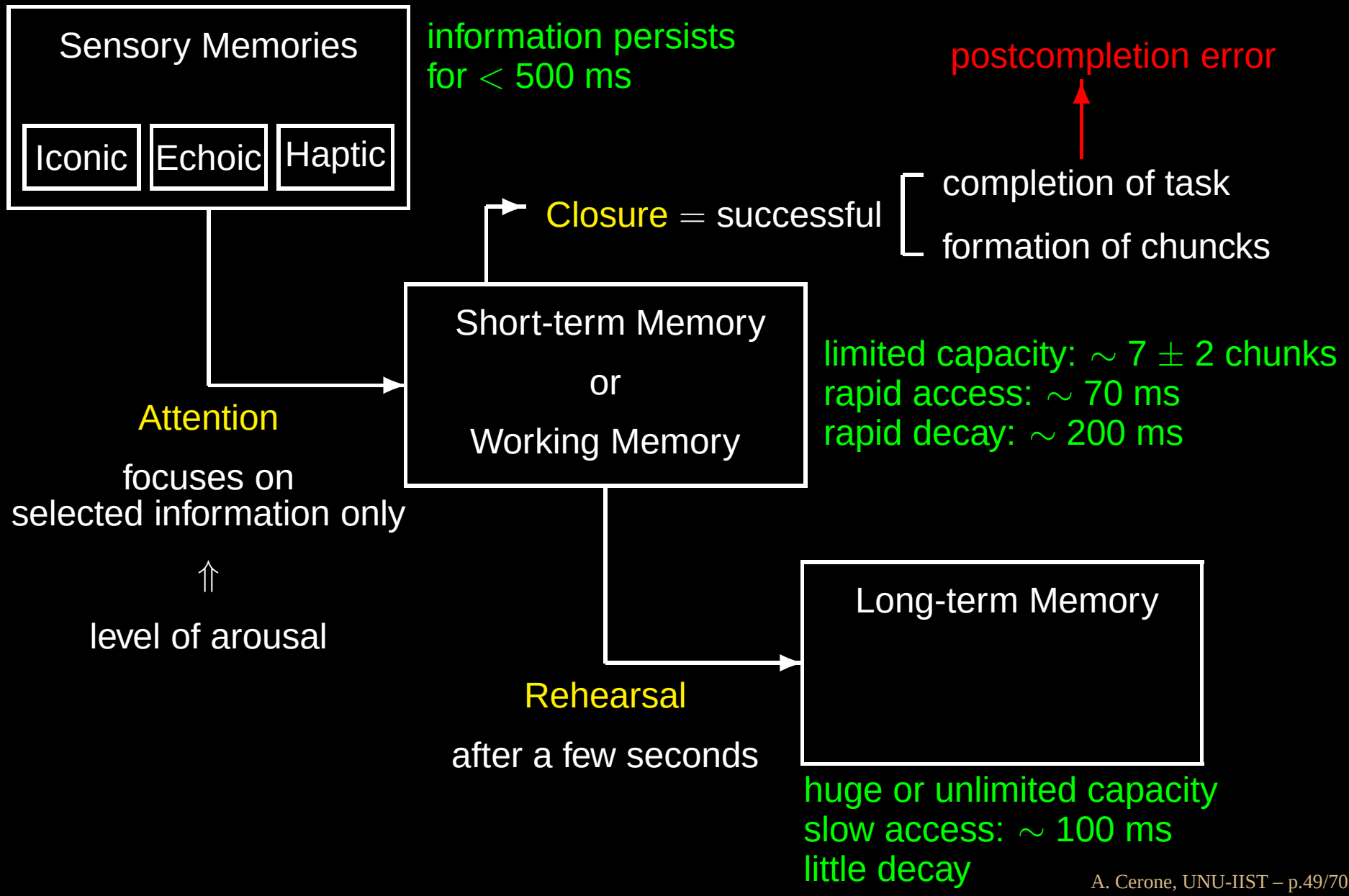
Human Memory



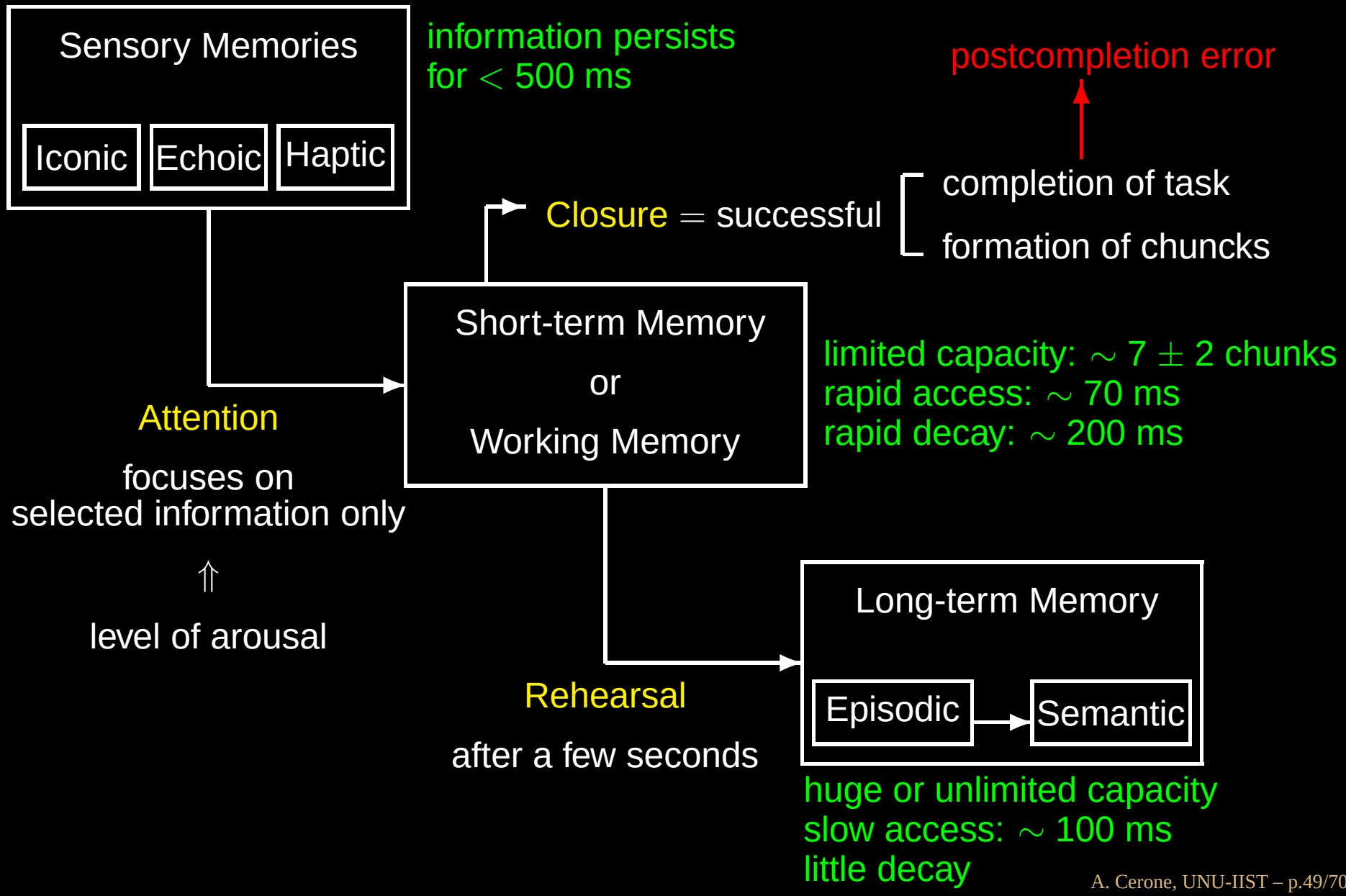
Human Memory



Human Memory



Human Memory



Modelling Human Behaviour

Why Poor Usability

- User friendly and easy to use from the point of view of the designer
- the designer is potentially a user $\Longrightarrow$
  - implicit assumptions on the user's capabilities and behaviour
  - explicit assumptions on the user's knowledge of the system — the user has entirely read and understood the manual
- interface viewed as plug-in separate from the rest of the system

Poor User Model

- User Model mimics the machine (user from the point of view of the designer)
- the designer is potentially a user $\Longrightarrow$
  - implicit assumptions: user remembers all required actions correctly
  - explicit assumptions on the user's knowledge of the system: none
- interface viewed as plug-in separate from the rest of the system: first modeled machine and then the user

Improving Usability

- USER = first priority in the requirements of interactive systems (SE)
- study of the mind (perception, thinking and learning) and behaviour of the human being (Psychology) and related experiments
- explicit assumptions on user's physical and cognitive limitations and environmental and social constraints (Ergonomics, Cognitive Science and Sociology)
- interface developed integrally with the rest of the system (SE) to support tasks people want to do and prevent / forgive careless errors

ATM: Better User Model

- USER has an an explicit goal and performs a task (independent of a specific machine)
- user model should include cognitive aspects that explain the user's plausible behavior
- explicit assumptions on user's physical and cognitive limitations: short-term memory, attention, ...
- interface developed integrally with the rest of the system through an iterative modelling and analysis process to support tasks people want to do (e.g. authentication) and prevent / forgive careless errors (e.g. forgetting collection)

ATM: Goals and Tasks

ATM: Goals and Tasks

- Goals
  - get cash
  - get statement

ATM: Goals and Tasks

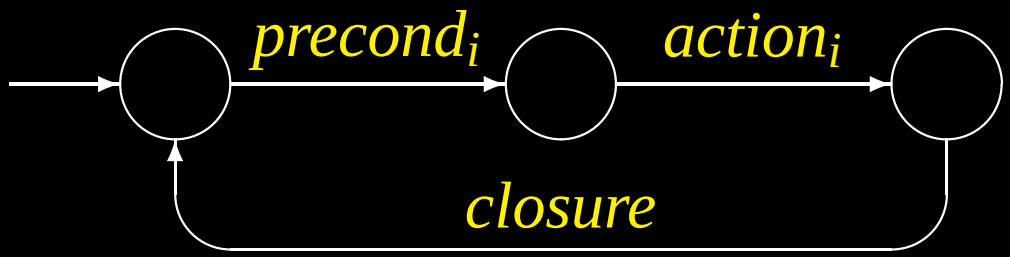
- Goals
  - get cash
  - get statement
- Tasks
  - top-level task
  - get authenticated (with the right balance between security and usability)
  - collect everything, either cash and card or statement and card (in any order)

ATM: Goals and Tasks

- Goals
  - get cash
  - get statement
- Tasks
  - top-level task
  - get authenticated (with the right balance between security and usability)
  - collect everything, either cash and card or statement and card (in any order)
  - basic task: single action (enter pin, … , collect cash)

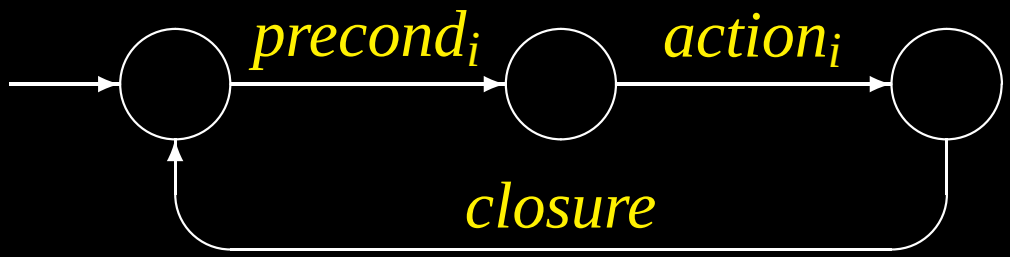
Goals, Actions, Closure

Goal action

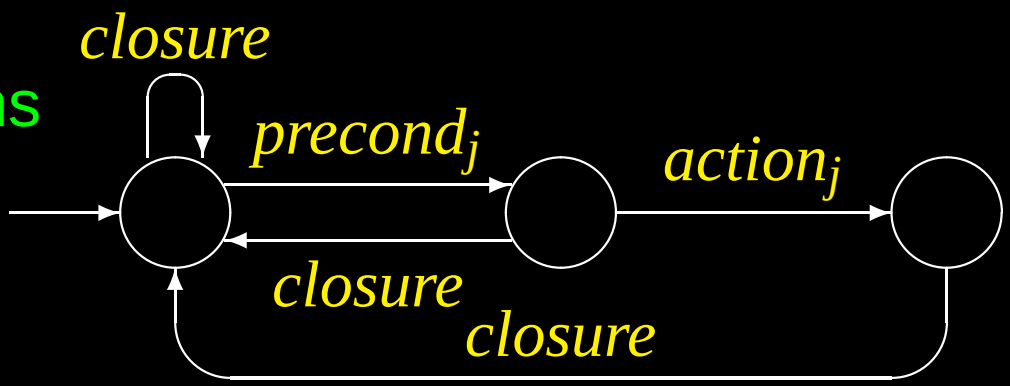


Goals, Actions, Closure

Goal action

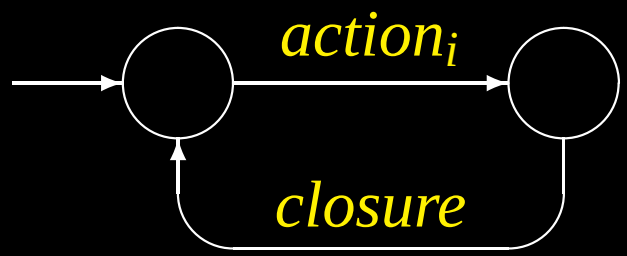


Other actions

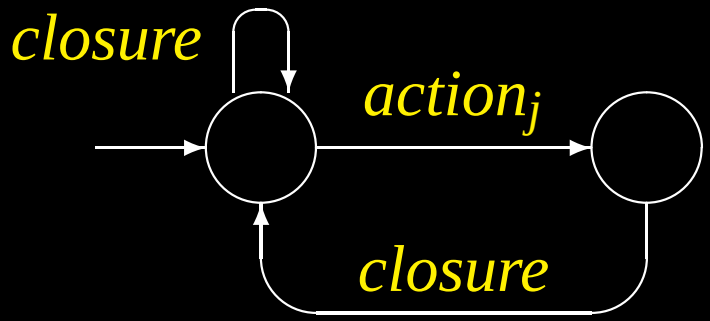


Goals, Actions, Closure

Goal action

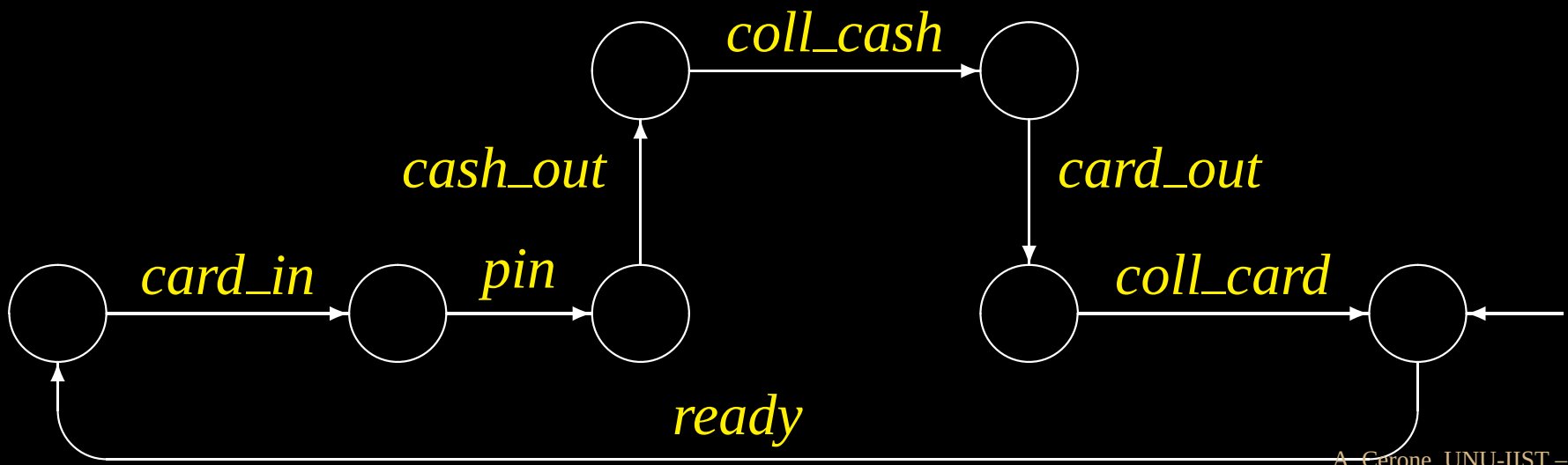


Other actions

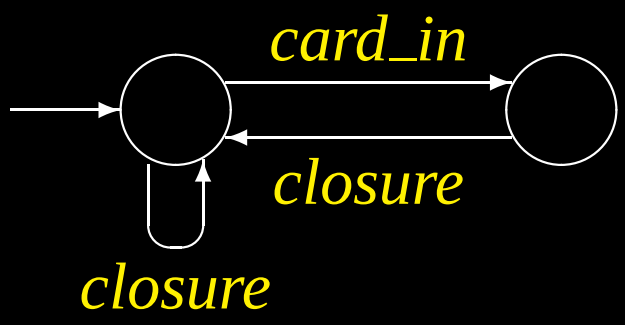


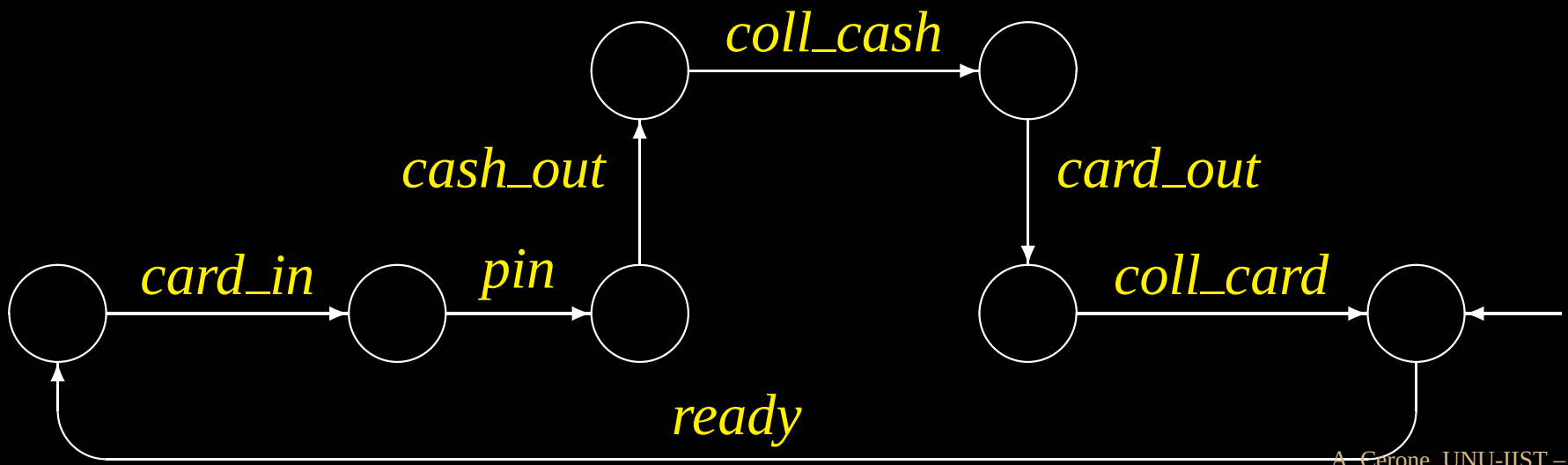
ATM Example Revisited

ATM: User

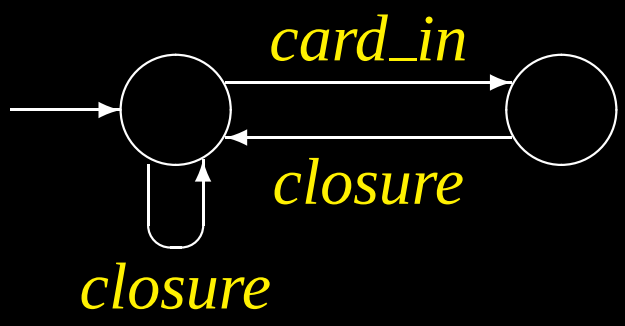


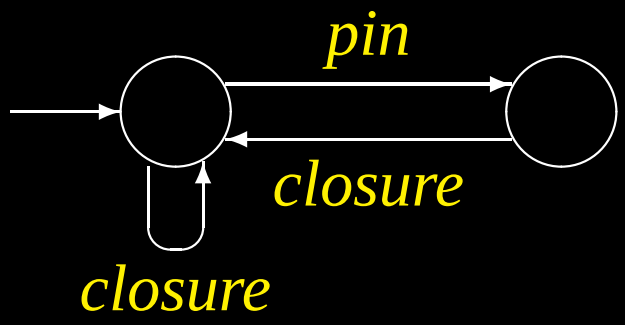
ATM: User

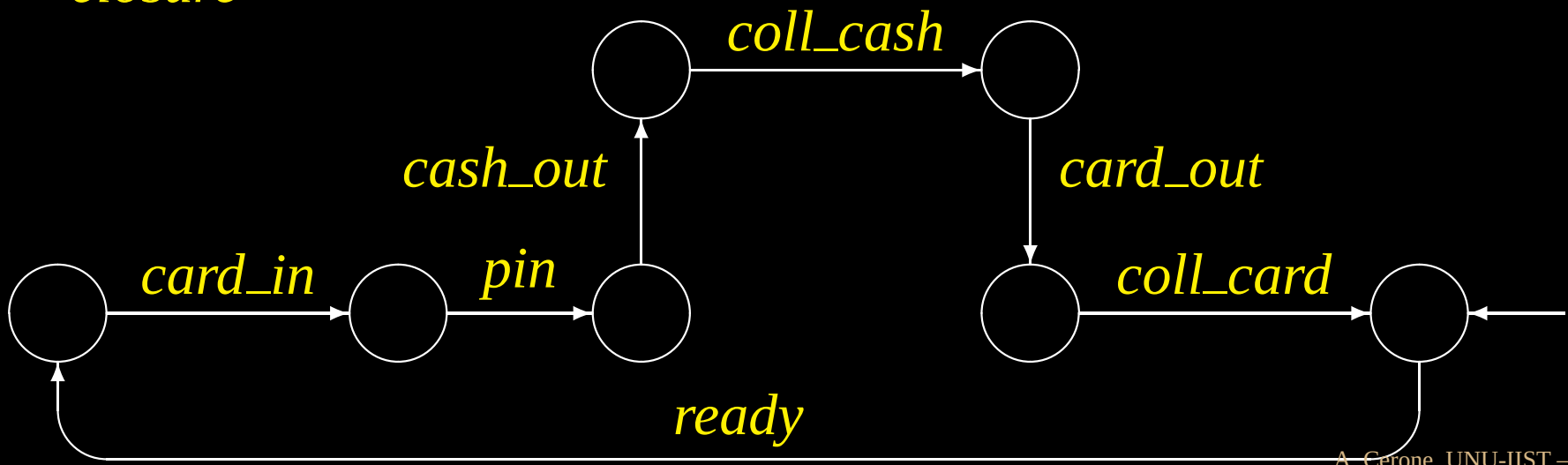

card_in
closure
closure


coll_cash
cash_out
card_out
card_in
pin
coll_card
ready

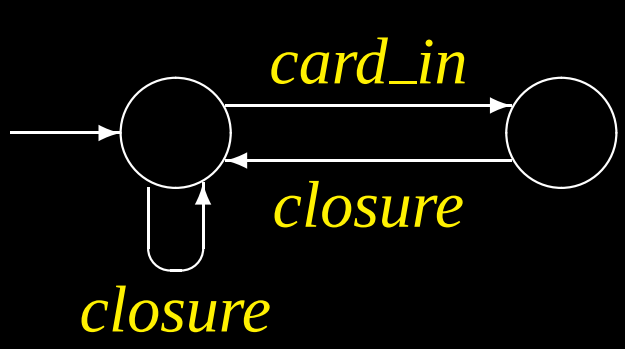
ATM: User

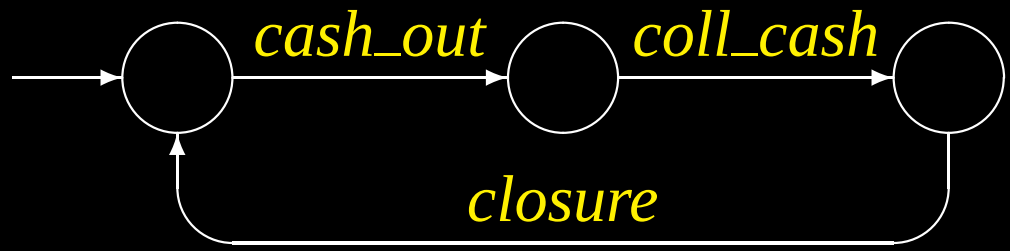

card_in
closure
closure

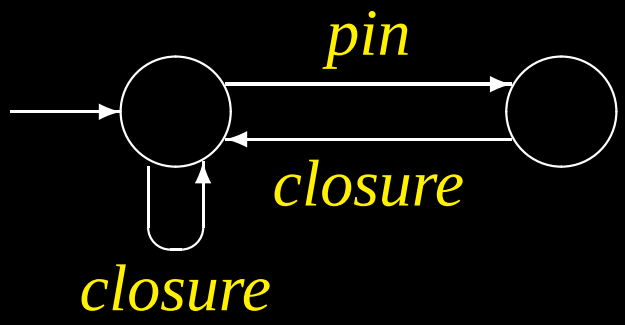

pin
closure
closure

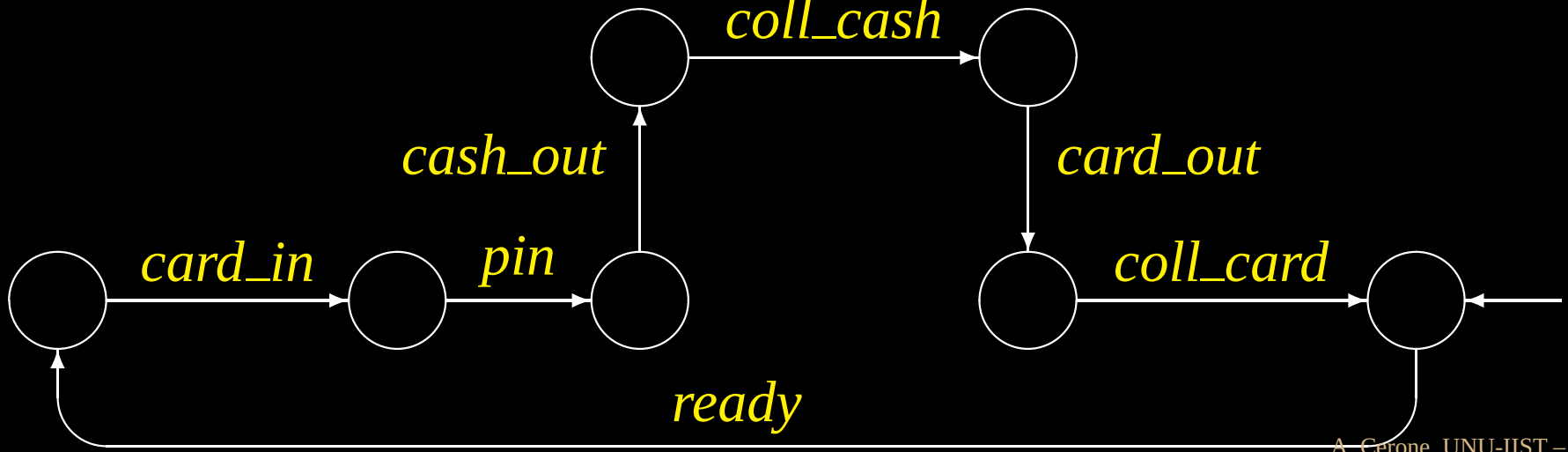

coll_cash
cash_out
card_out
card_in
pin
coll_card
ready

ATM: User

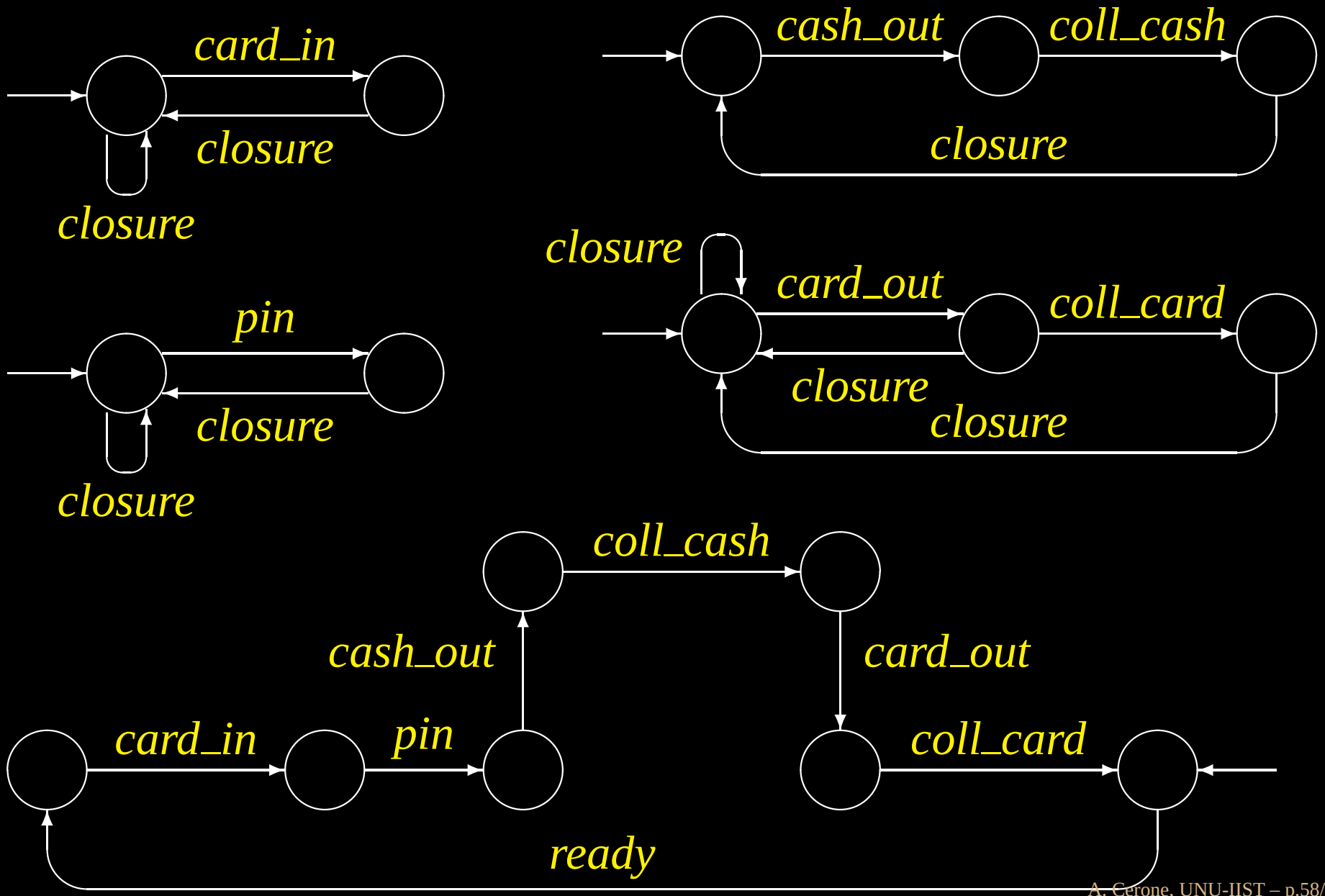

card_in
closure
closure


cash_out
coll_cash
closure

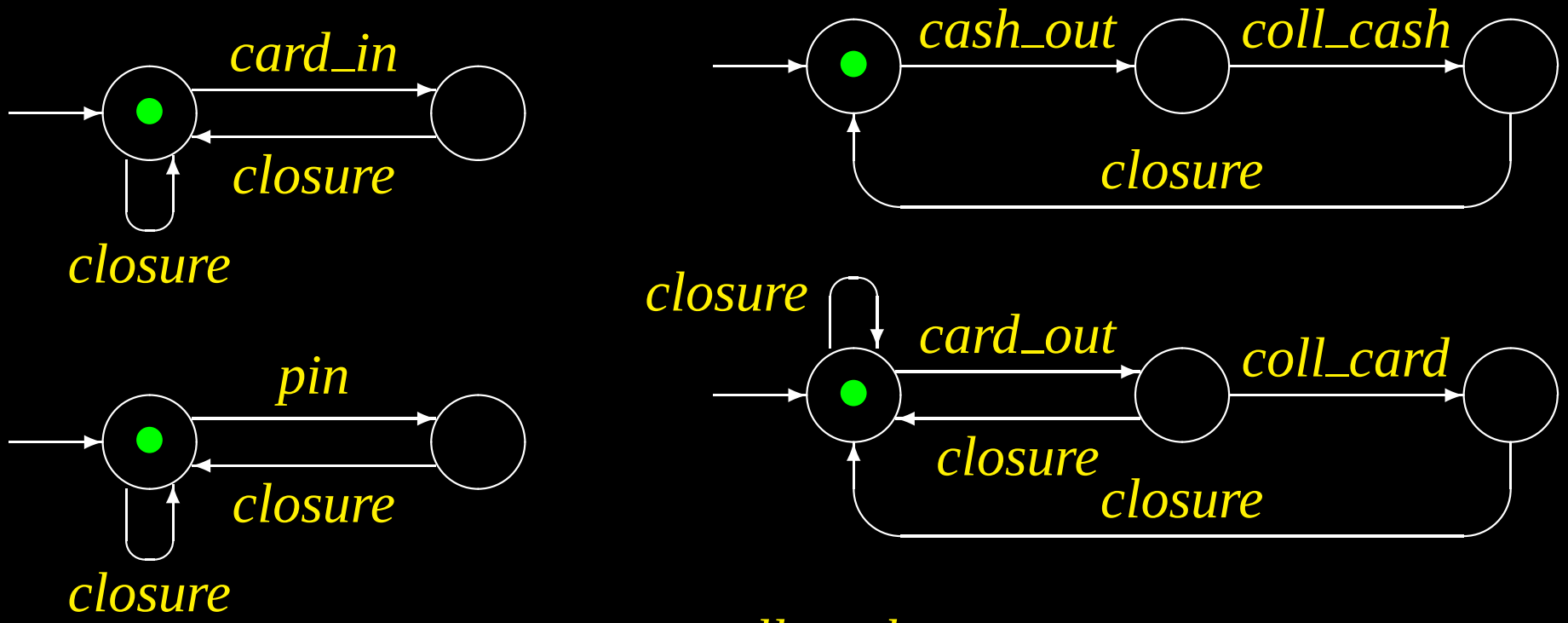

pin
closure
closure


coll_cash
cash_out
card_out
card_in
pin
coll_card
ready

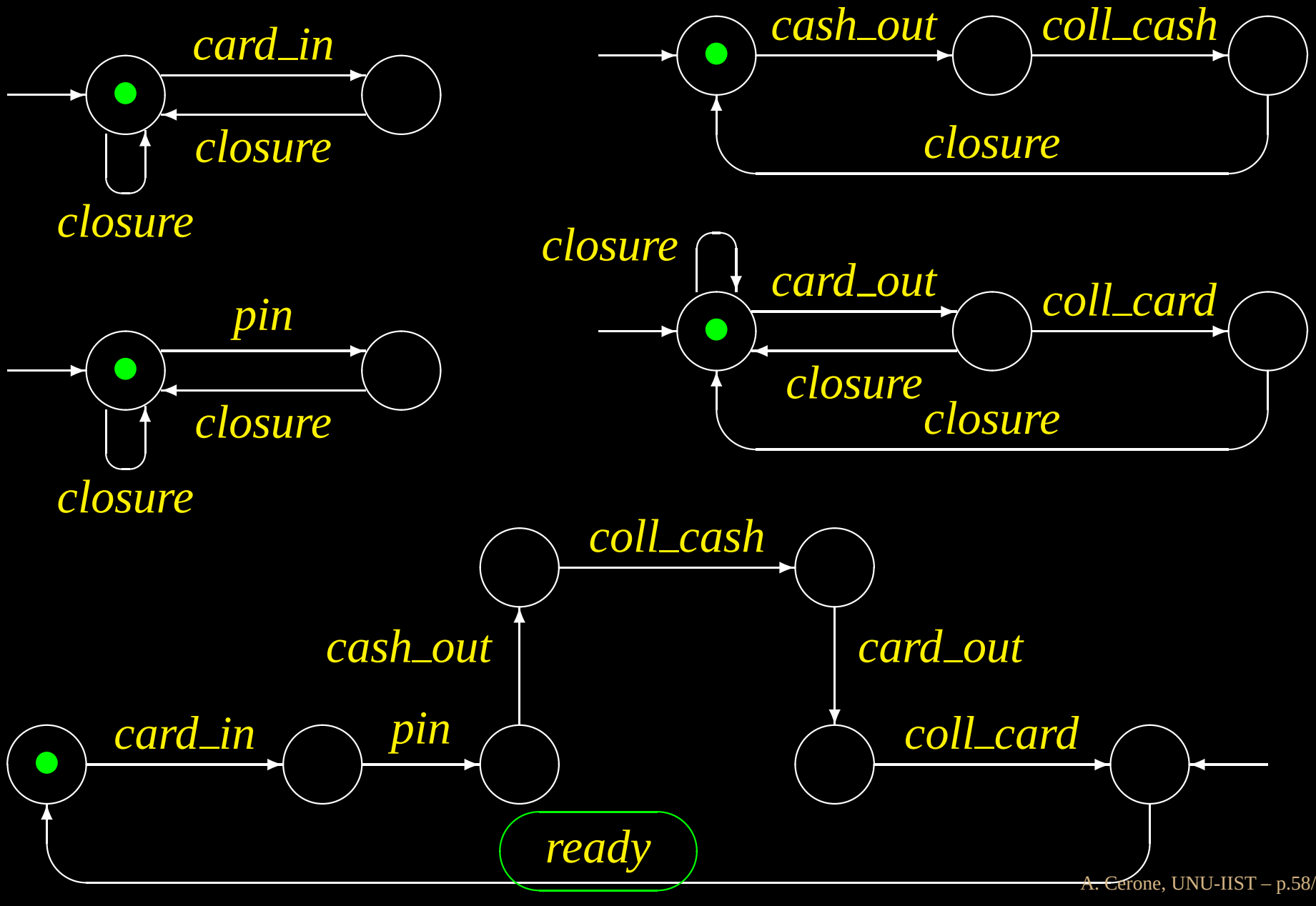
ATM: User


card_in
closure
closure
cash_out
coll_cash
closure
pin
closure
closure
closure
card_out
coll_card
closure
closure
coll_cash
cash_out
card_out
card_in
pin
coll_card
ready

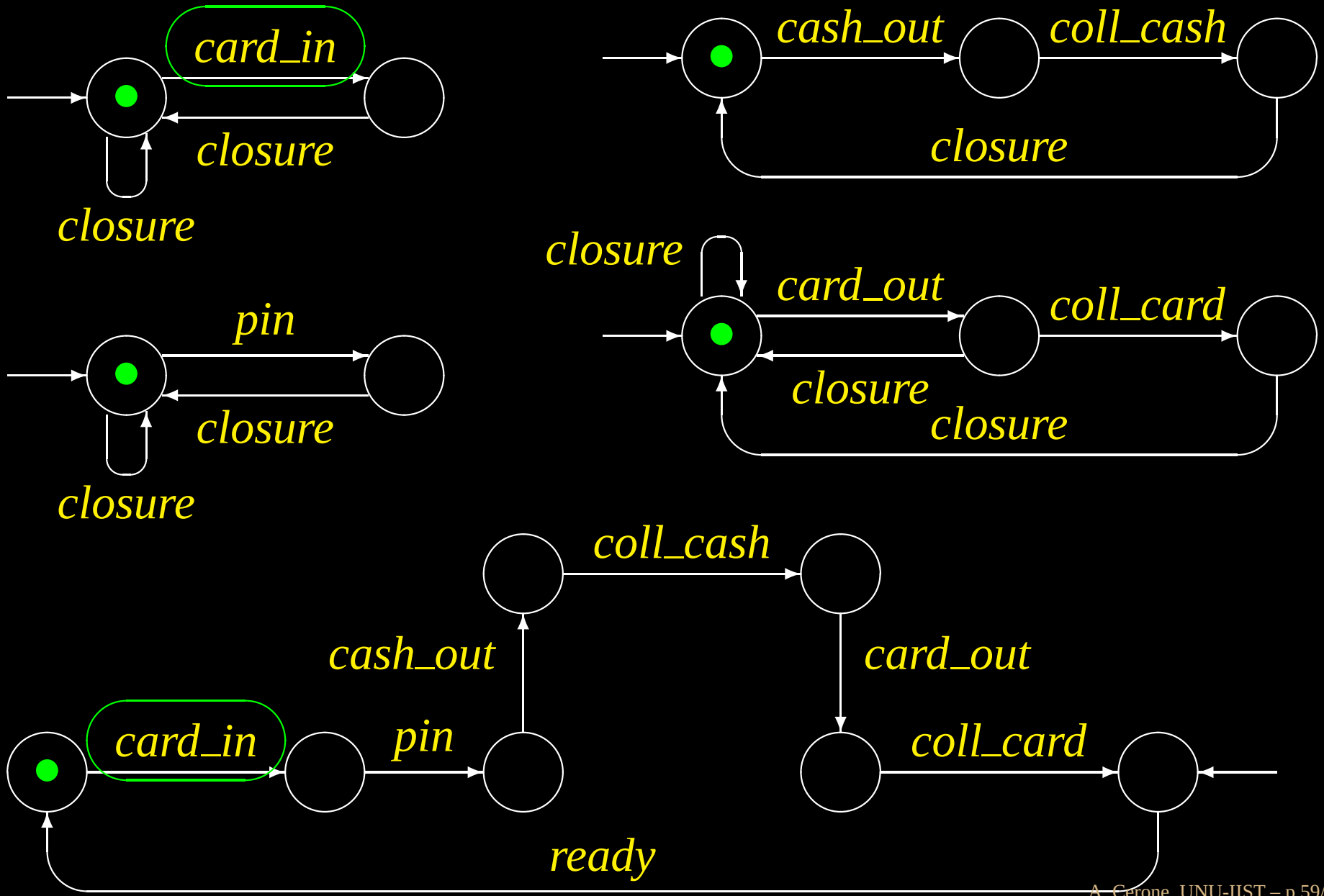
ATM: User


card_in
closure
closure
cash_out
coll_cash
closure
pin
closure
closure
closure
card_out
coll_card
closure
closure
coll_cash
cash_out
card_out
card_in
pin
coll_card
ready

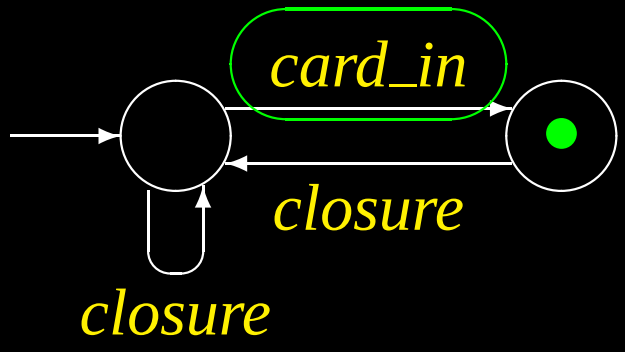
ATM: User

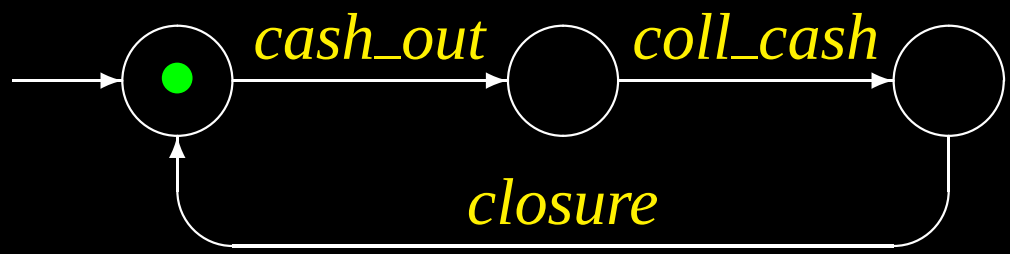

card_in
closure
closure
cash_out
coll_cash
closure
closure
card_out
coll_card
closure
closure
pin
closure
closure
coll_cash
cash_out
card_out
card_in
pin
coll_card
ready

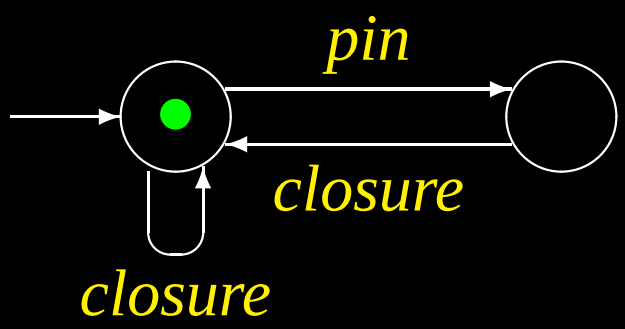
ATM: Interaction 1

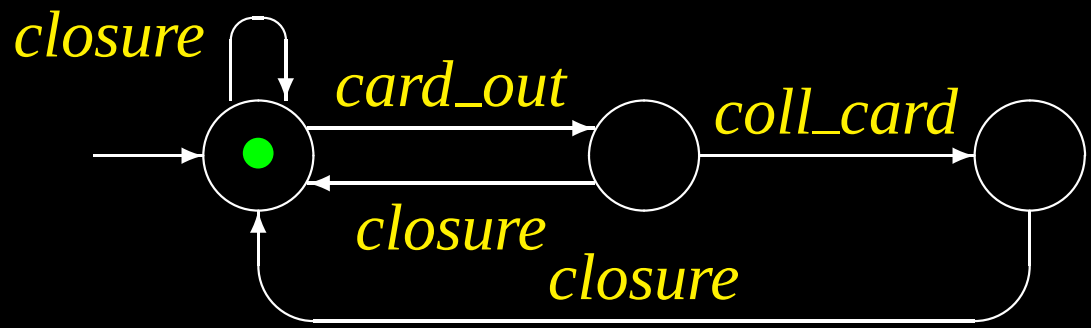


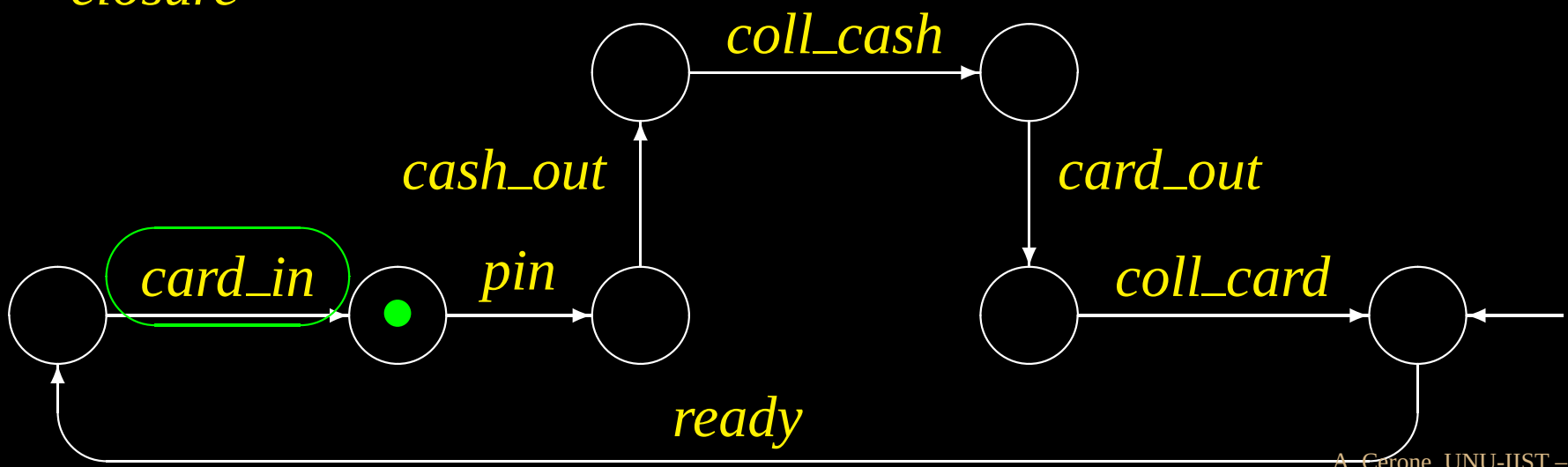
ATM: Interaction 1



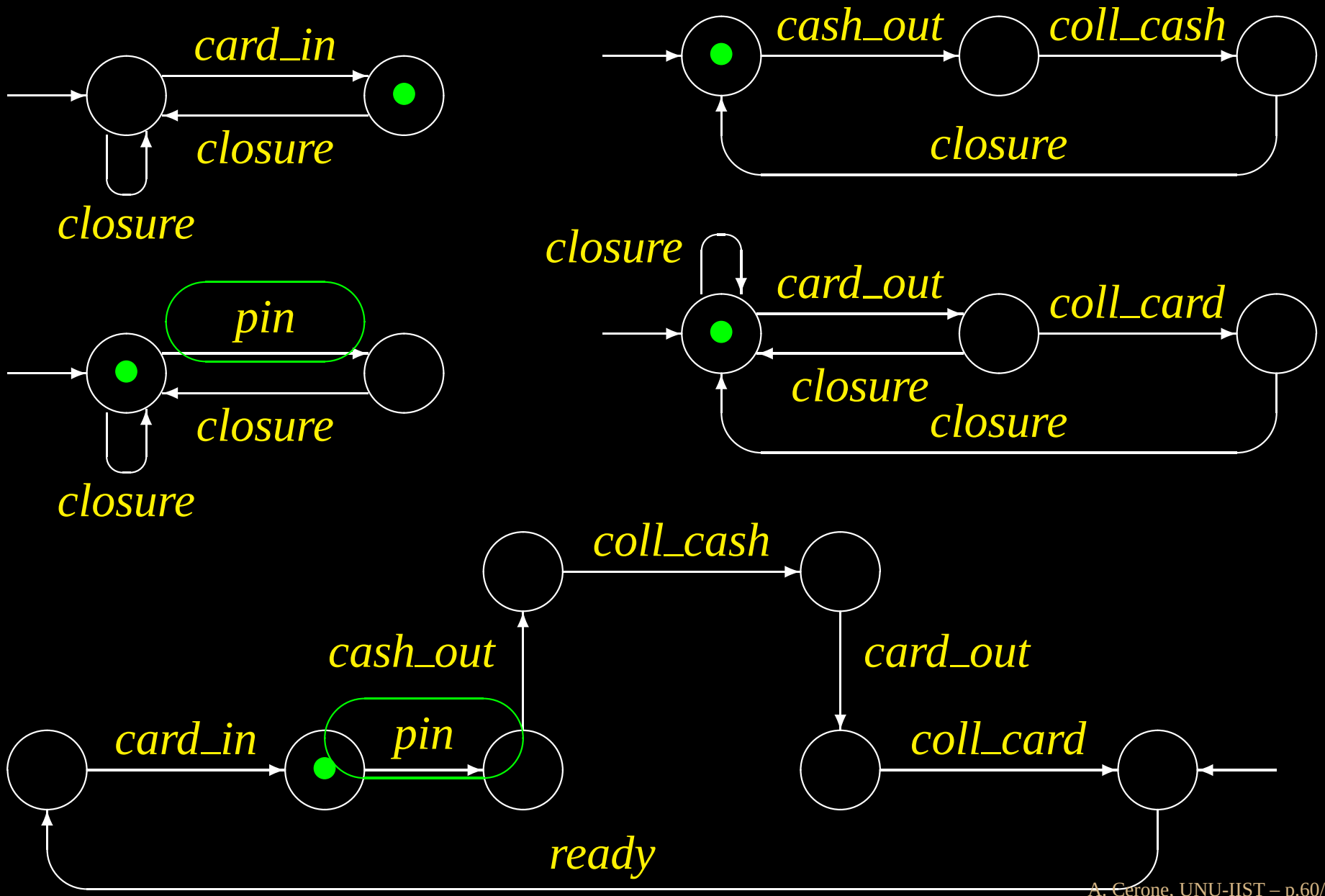




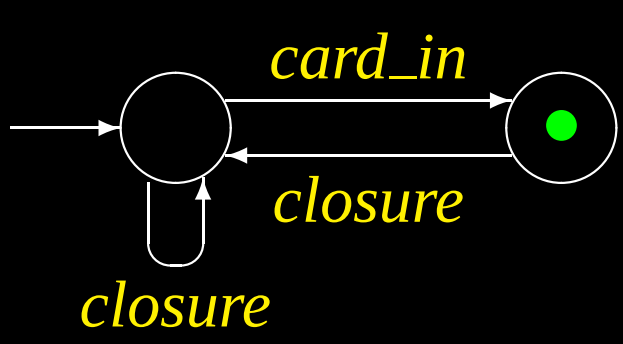


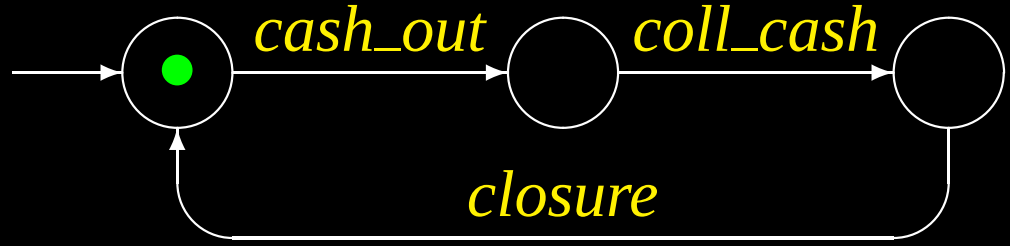


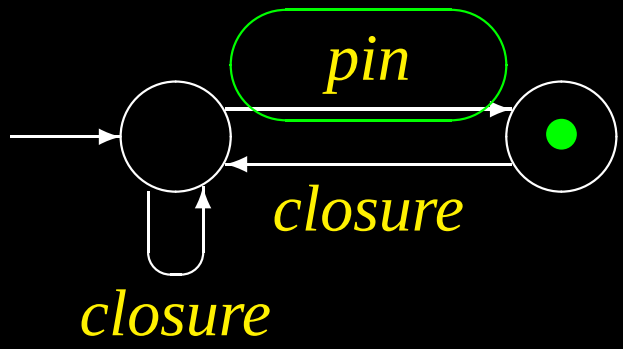
ATM: Interaction 2

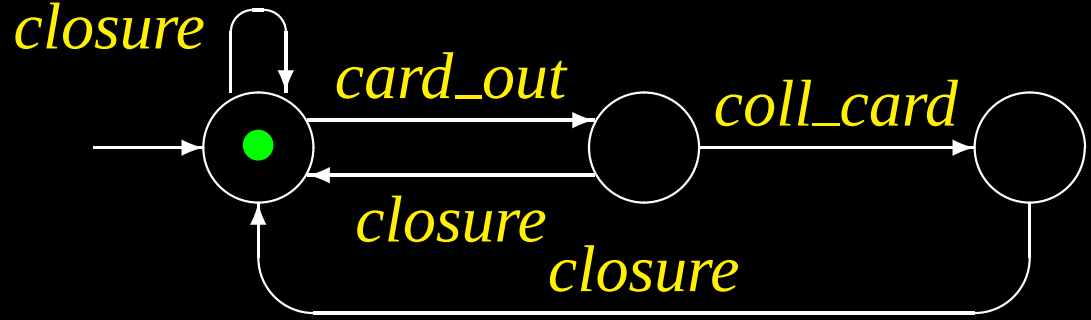


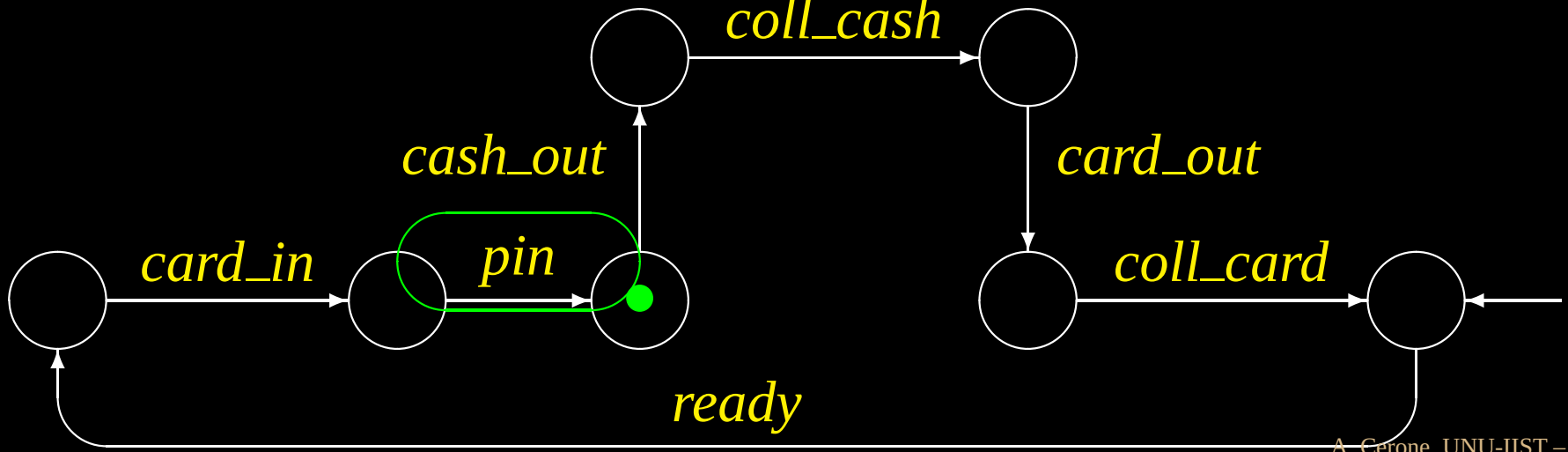
ATM: Interaction 2



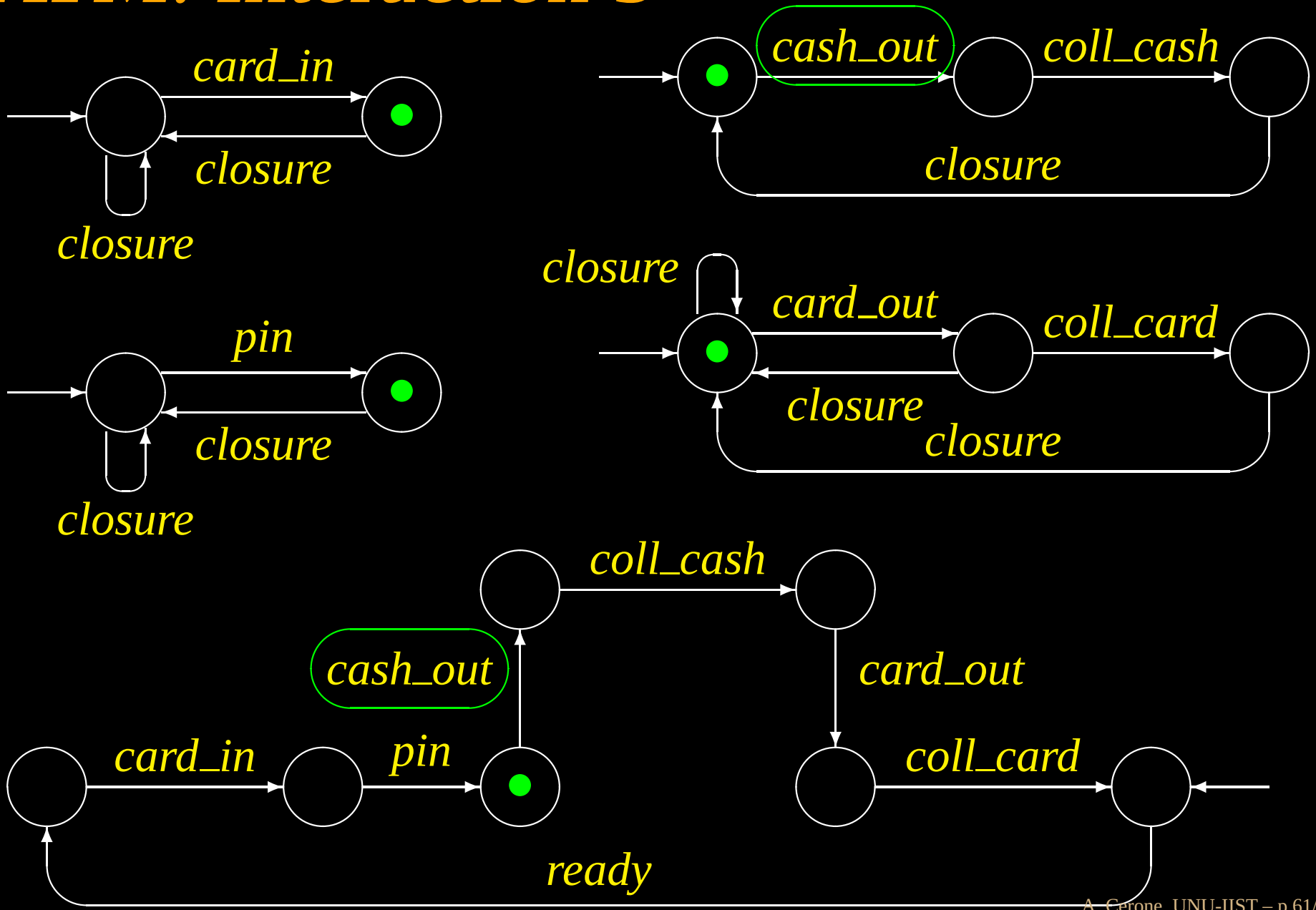




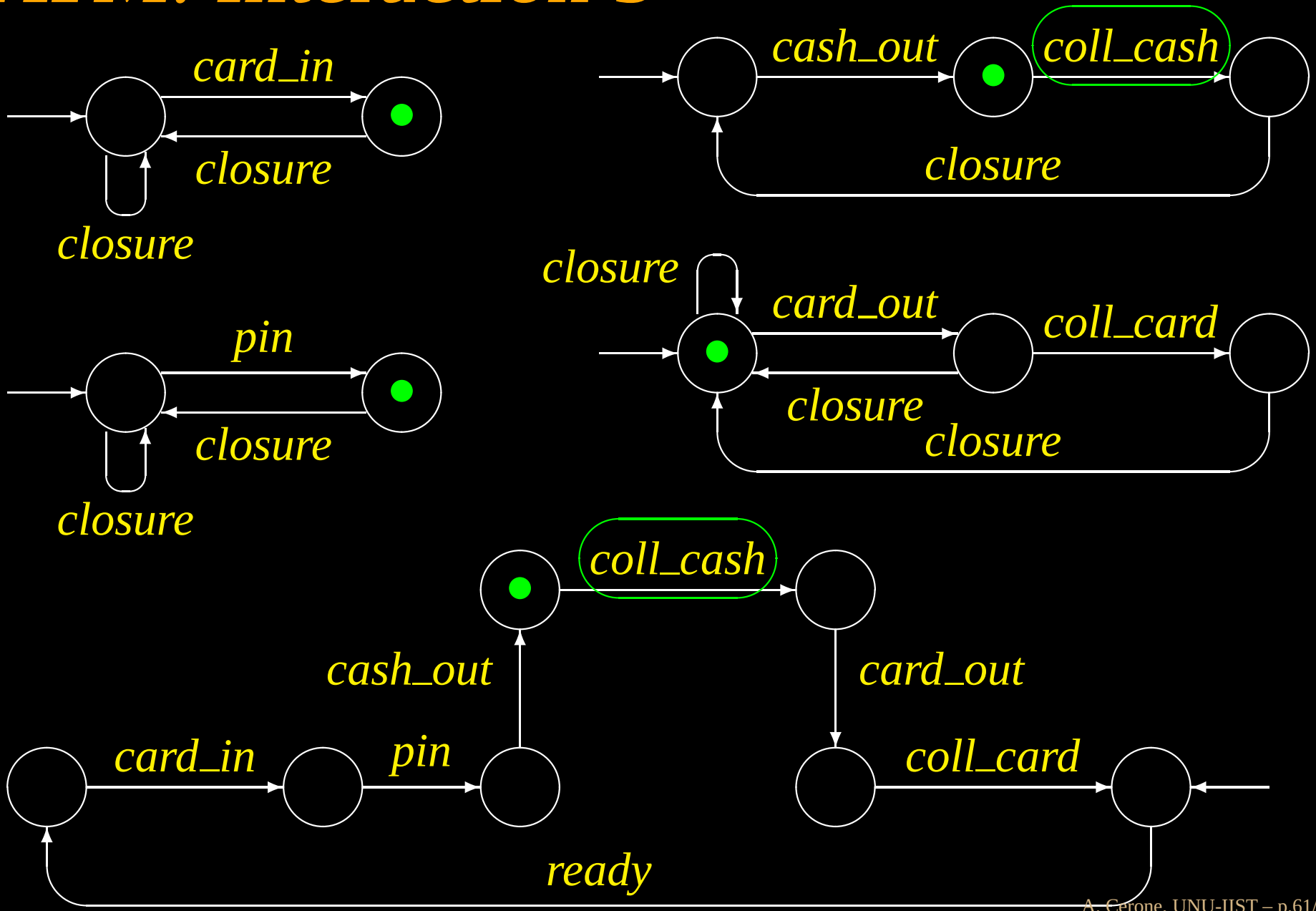




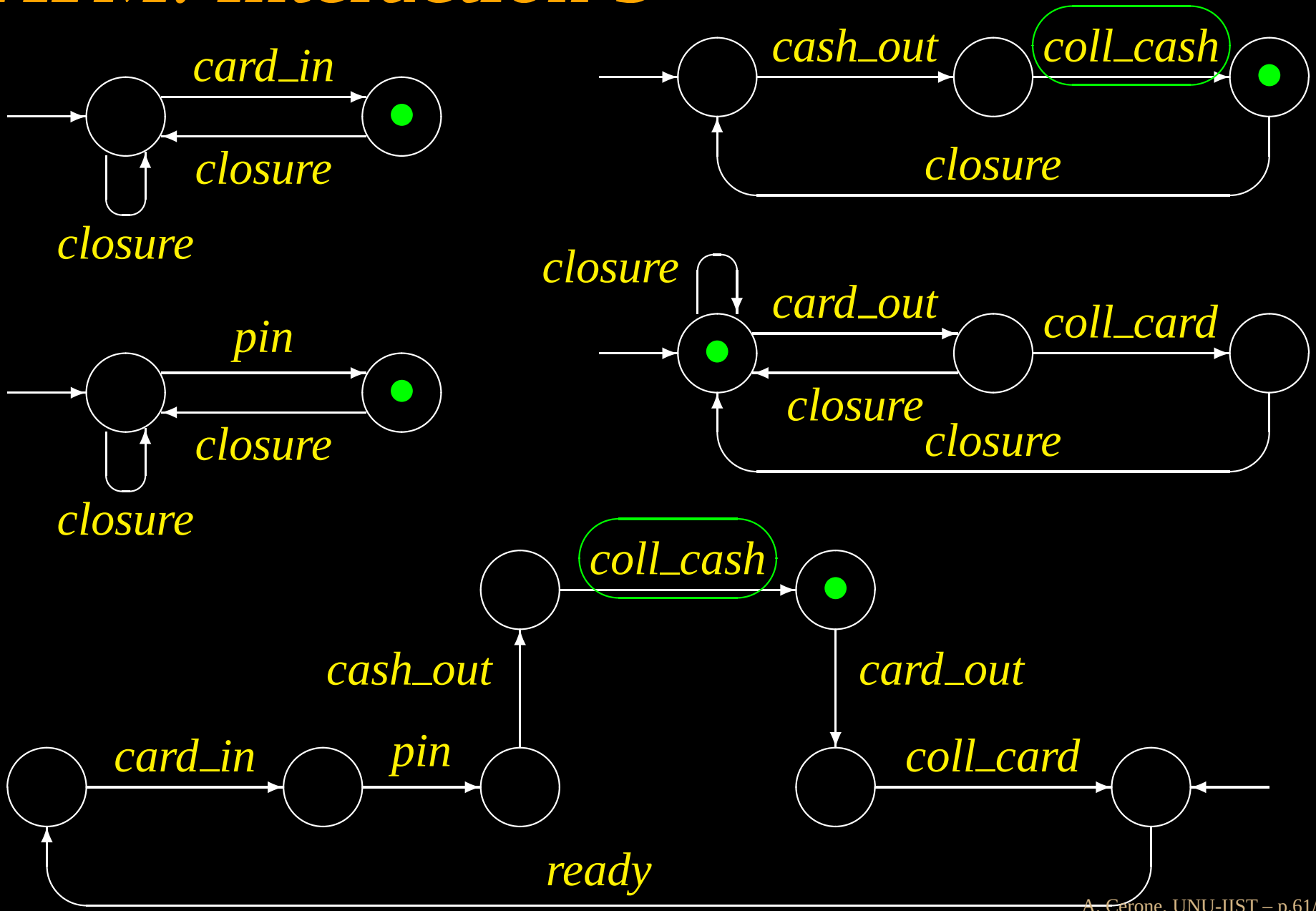
ATM: Interaction 3



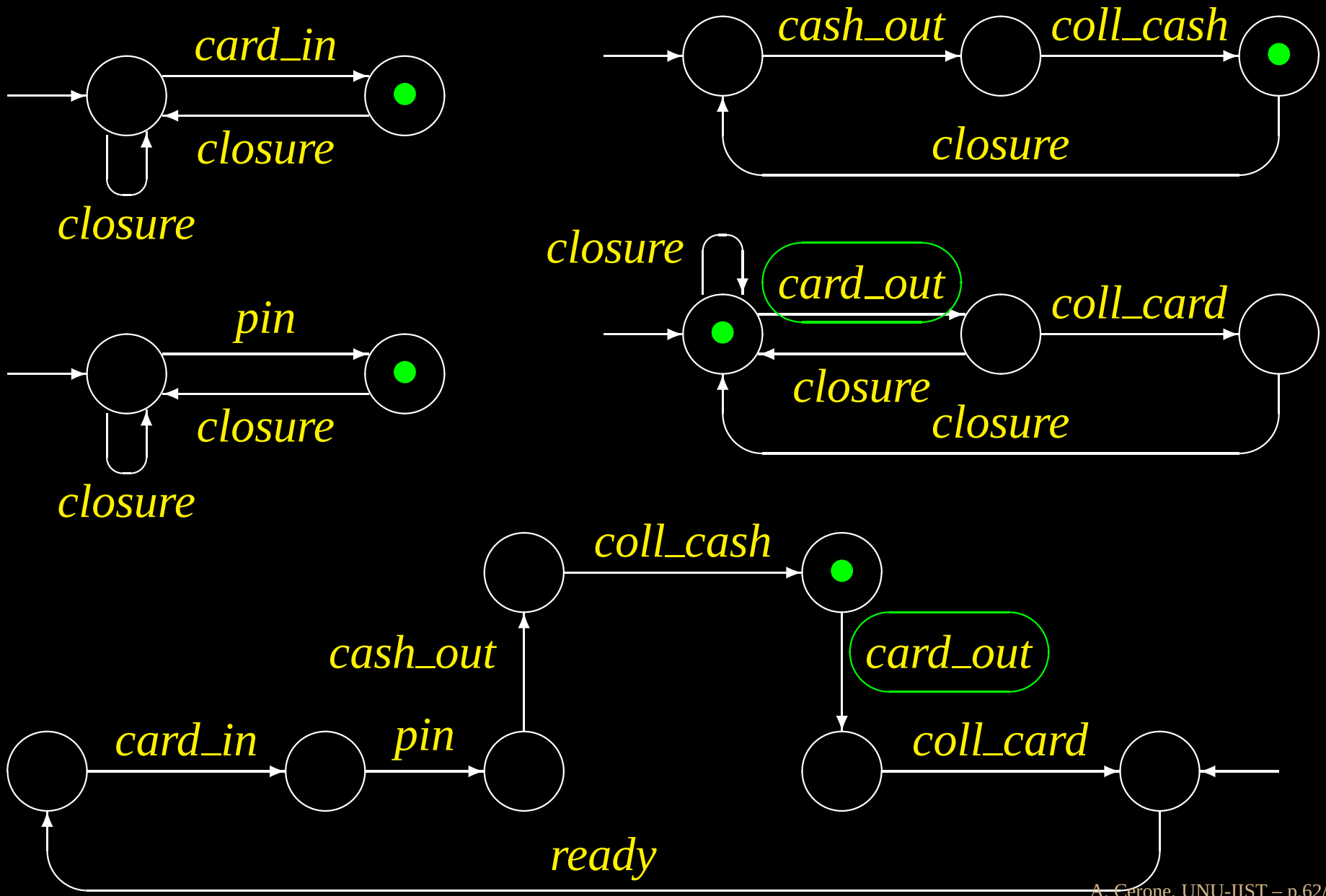
ATM: Interaction 3



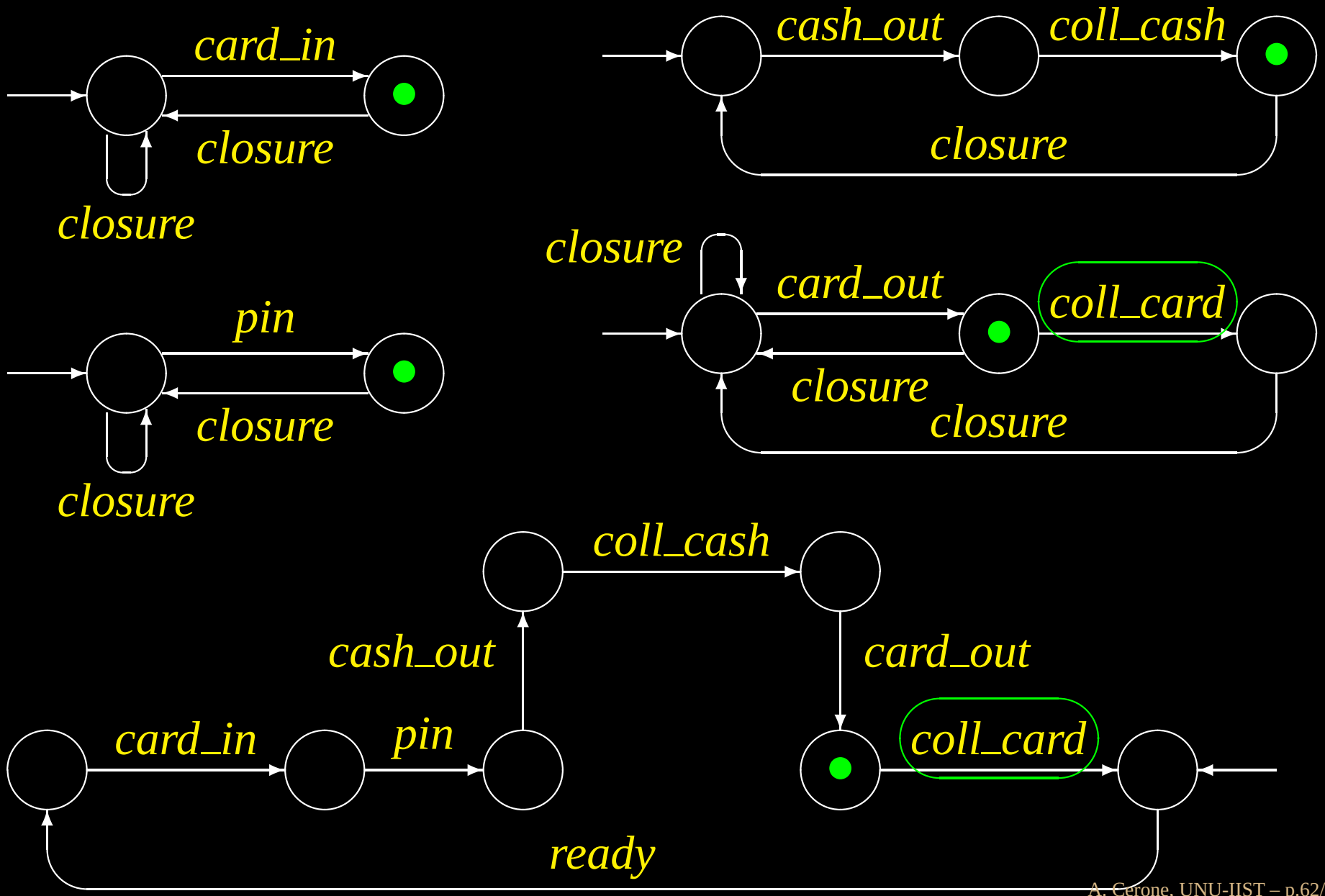
ATM: Interaction 3



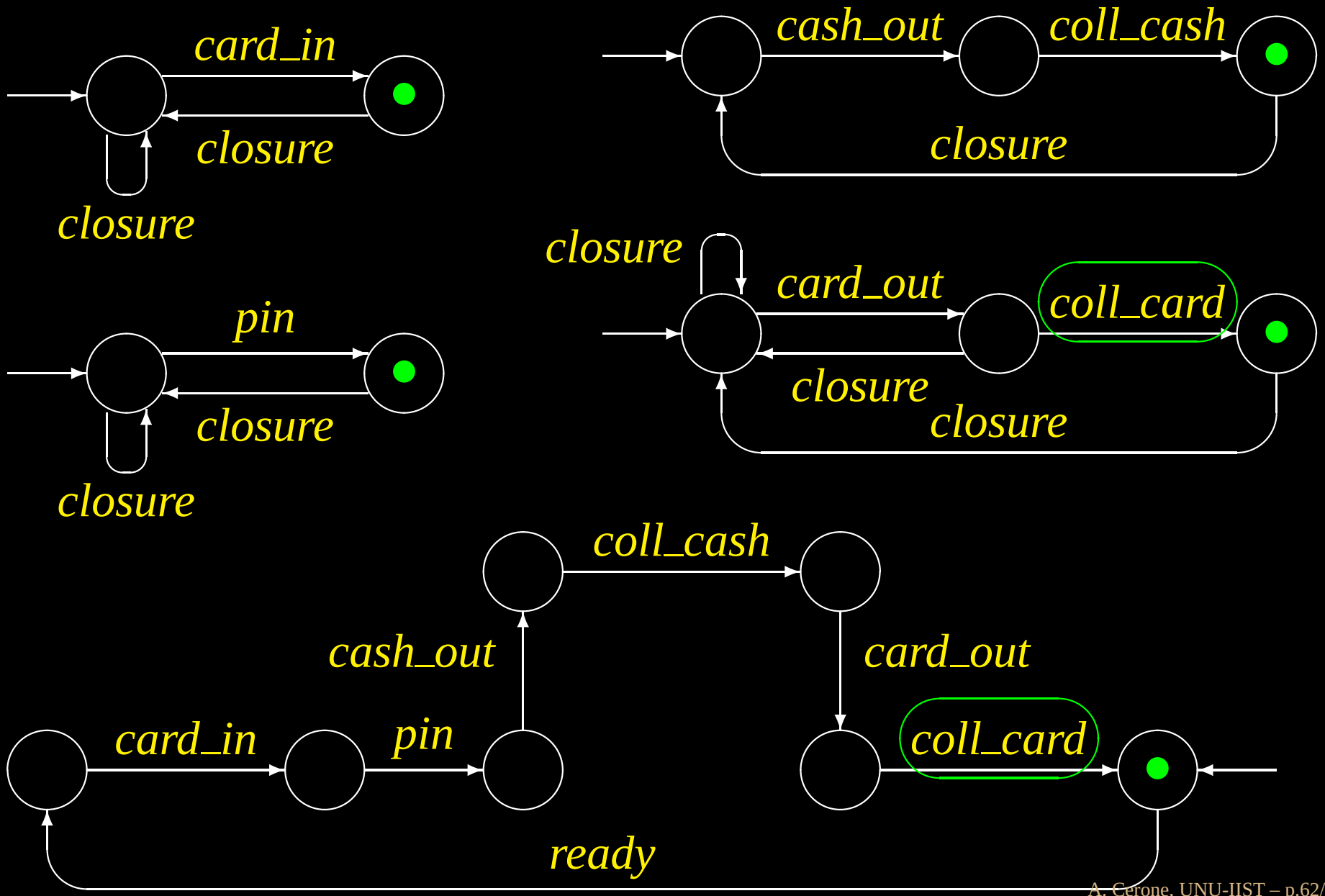
ATM: Interaction 4



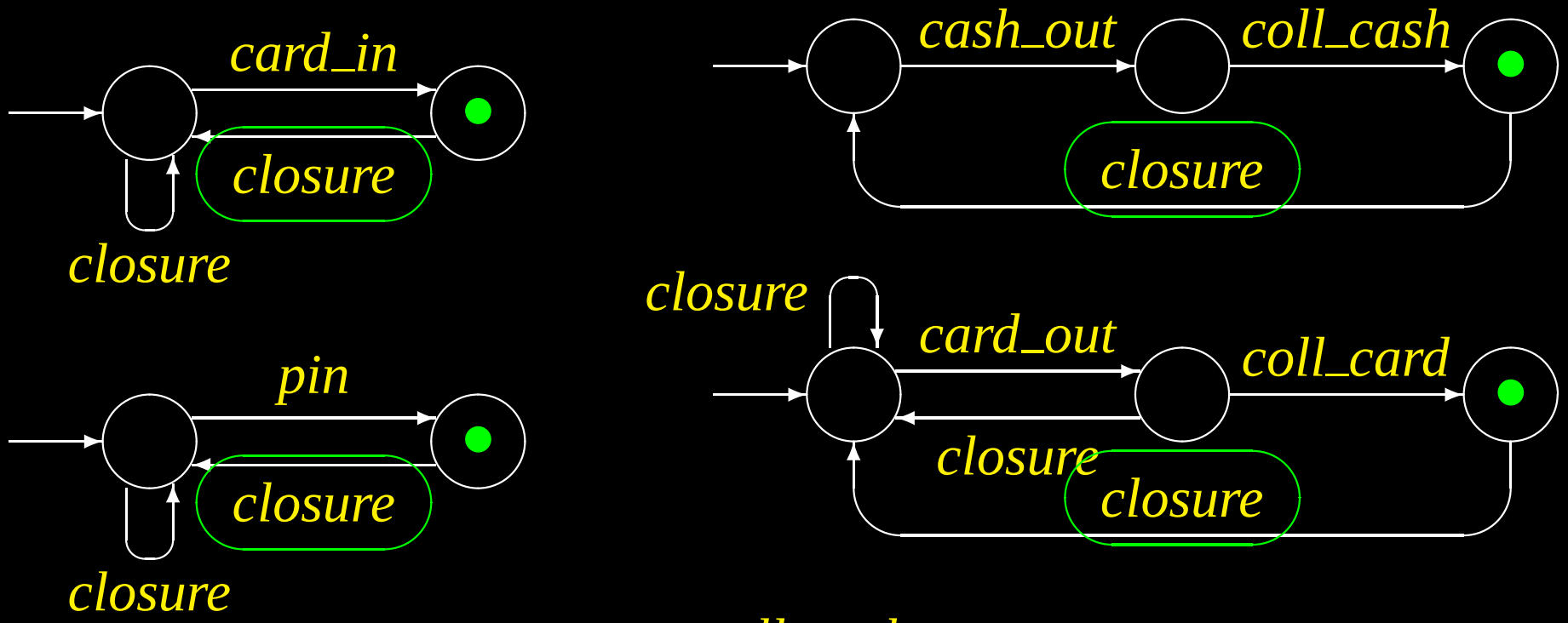
ATM: Interaction 4



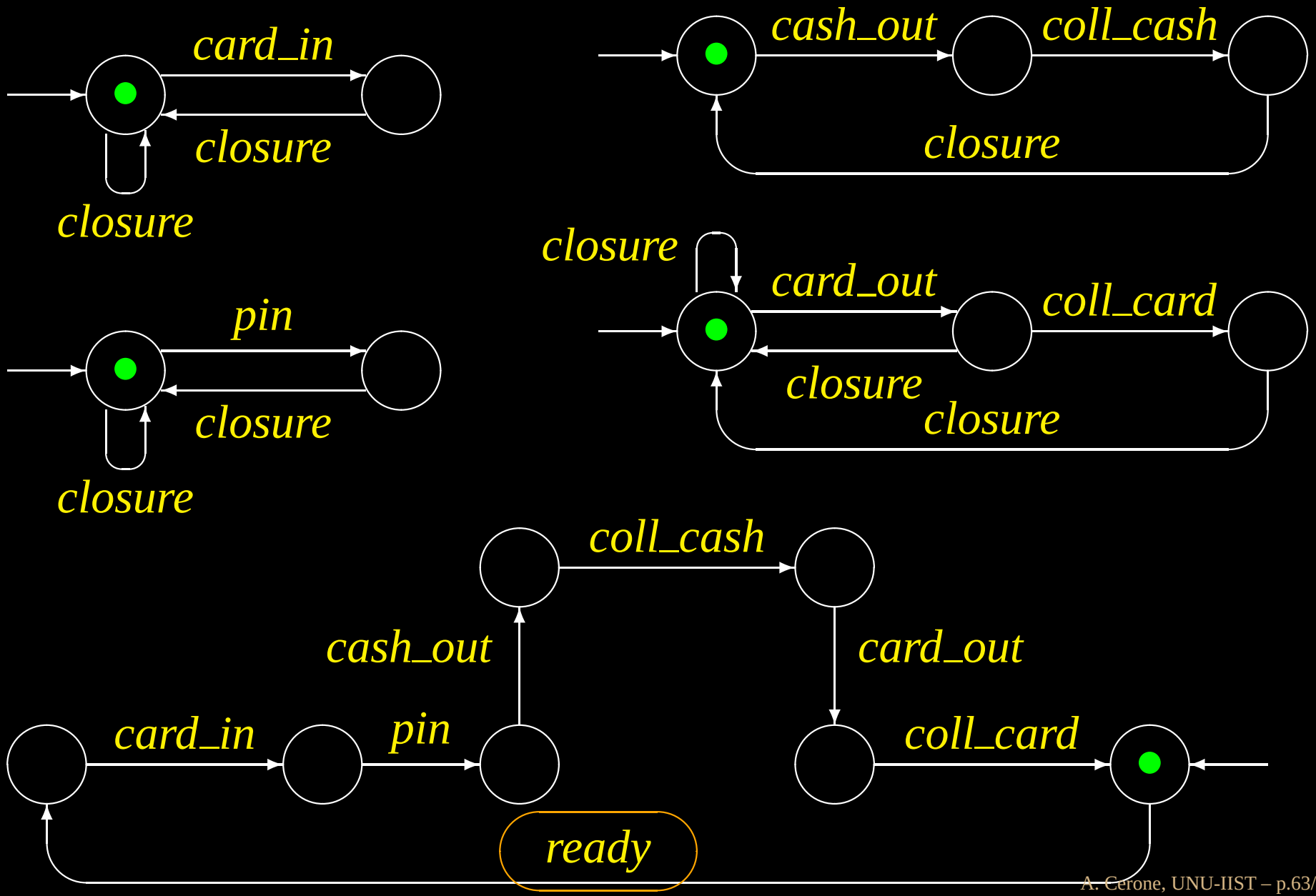
ATM: Interaction 4



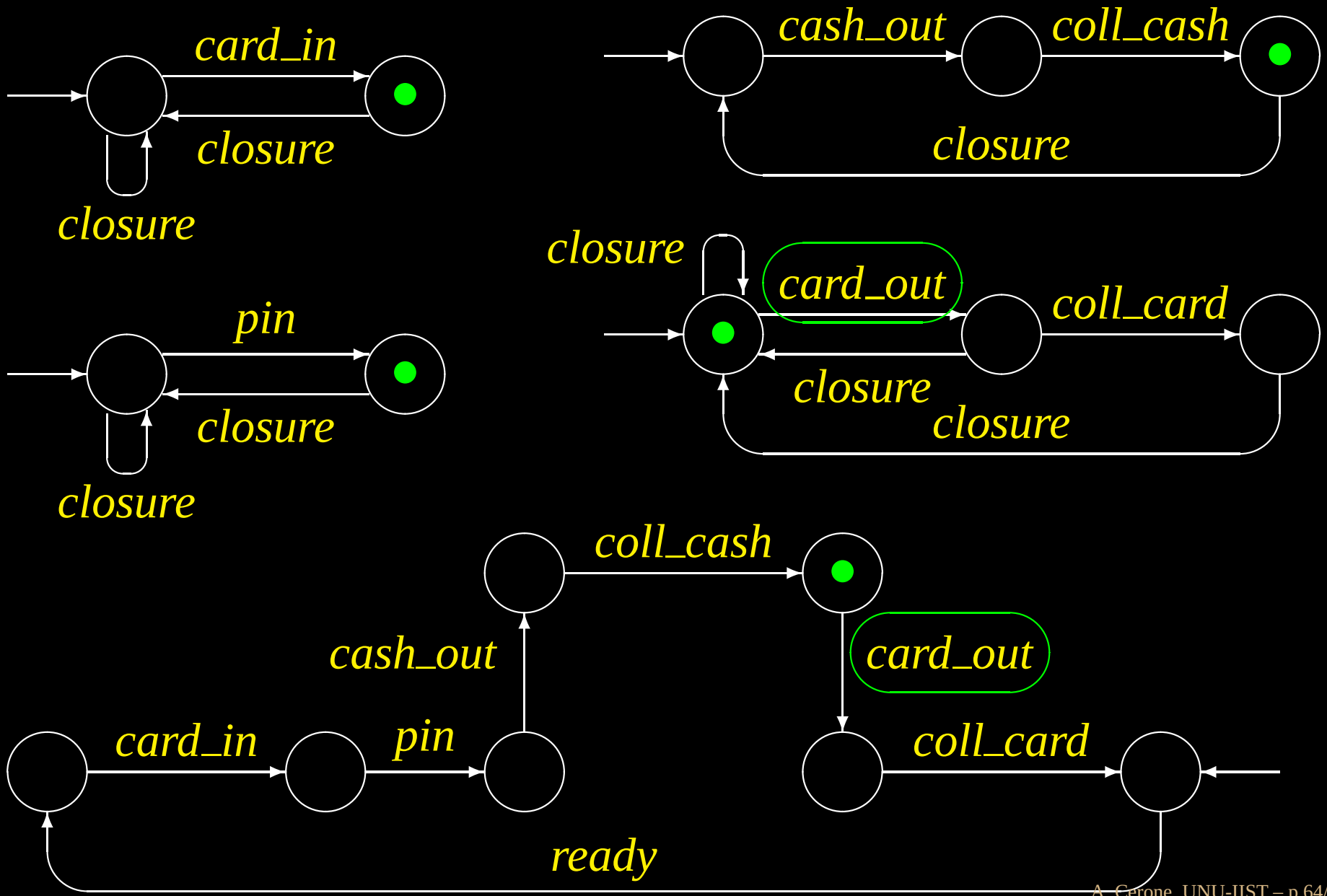
ATM: Closure


card_in
closure
closure
cash_out
coll_cash
closure
closure
card_out
coll_card
closure
closure
pin
closure
closure
coll_cash
cash_out
card_out
card_in
pin
coll_card
ready

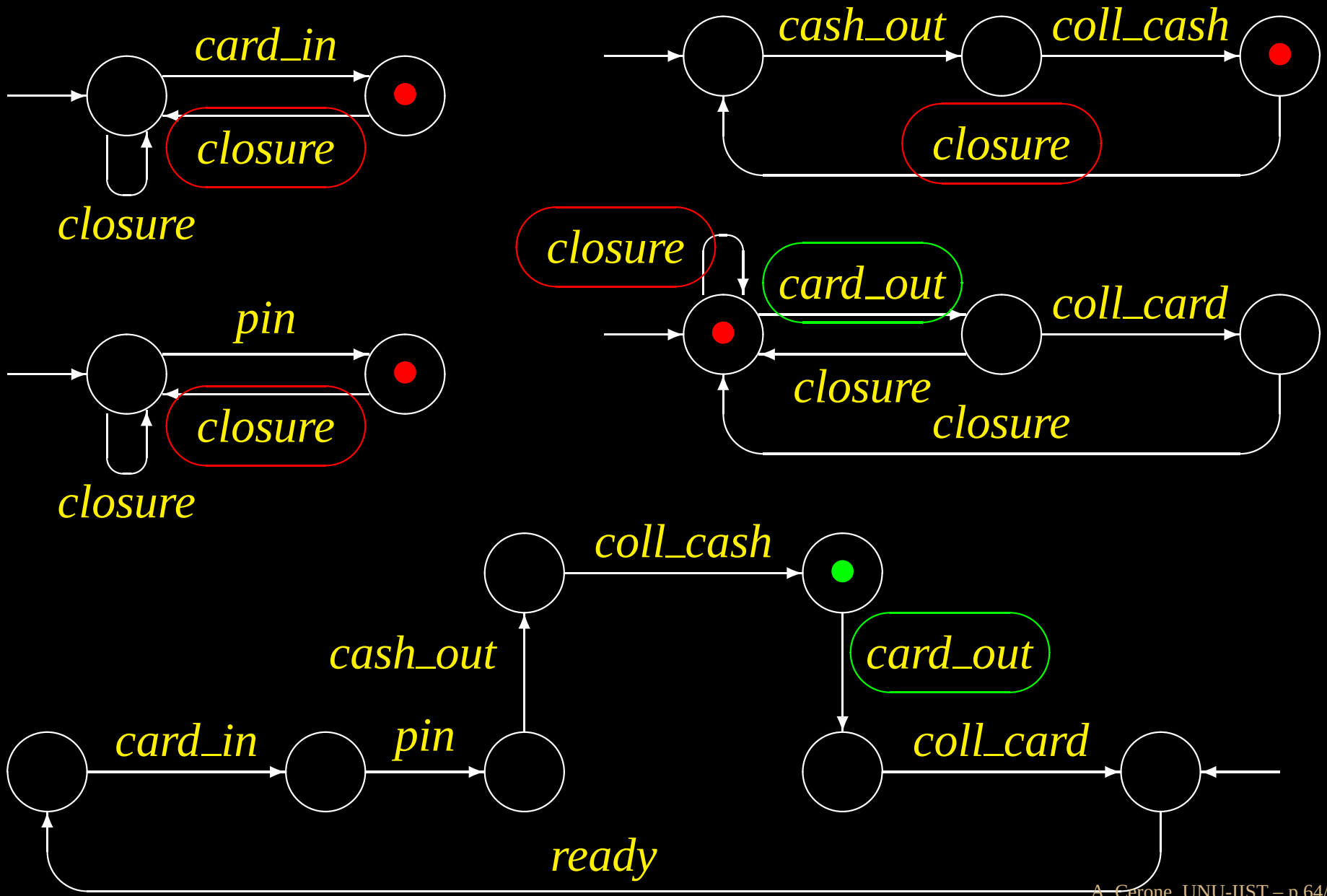
ATM: Closure


card_in
closure
closure
cash_out
coll_cash
closure
pin
closure
closure
closure
card_out
coll_card
closure
closure
coll_cash
cash_out
card_out
card_in
pin
coll_card
ready

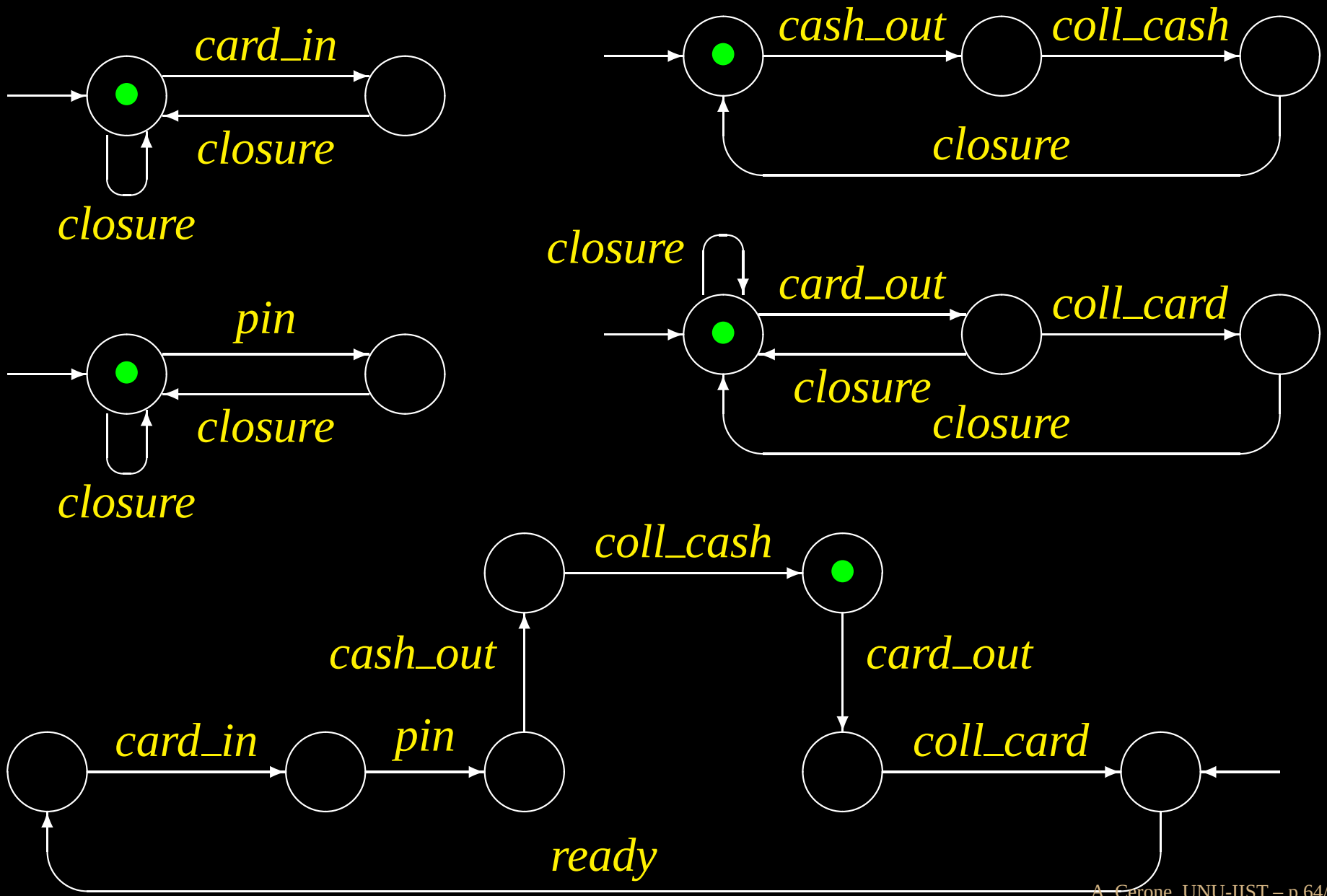
ATM: Post-completion Error



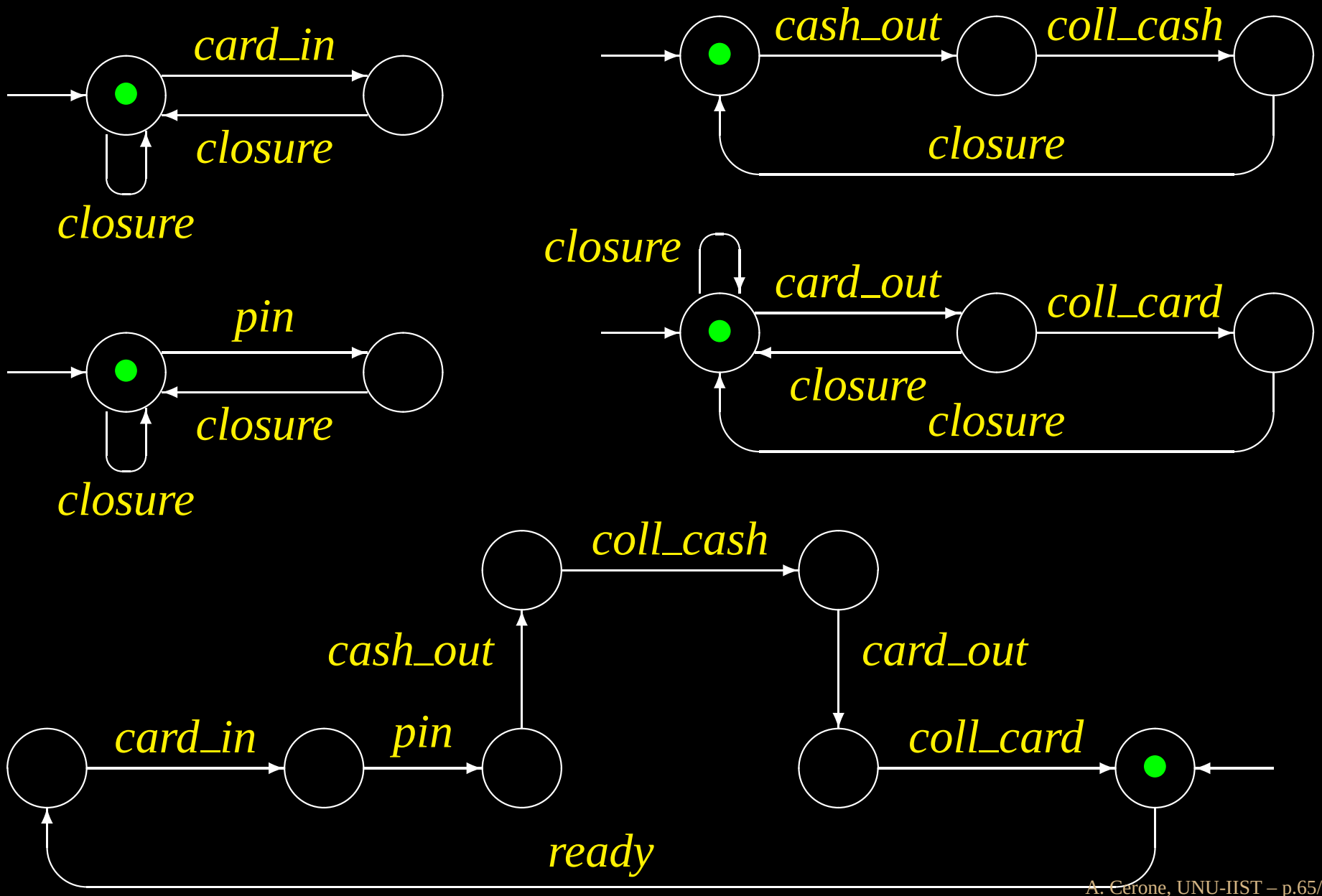
ATM: Post-completion Error



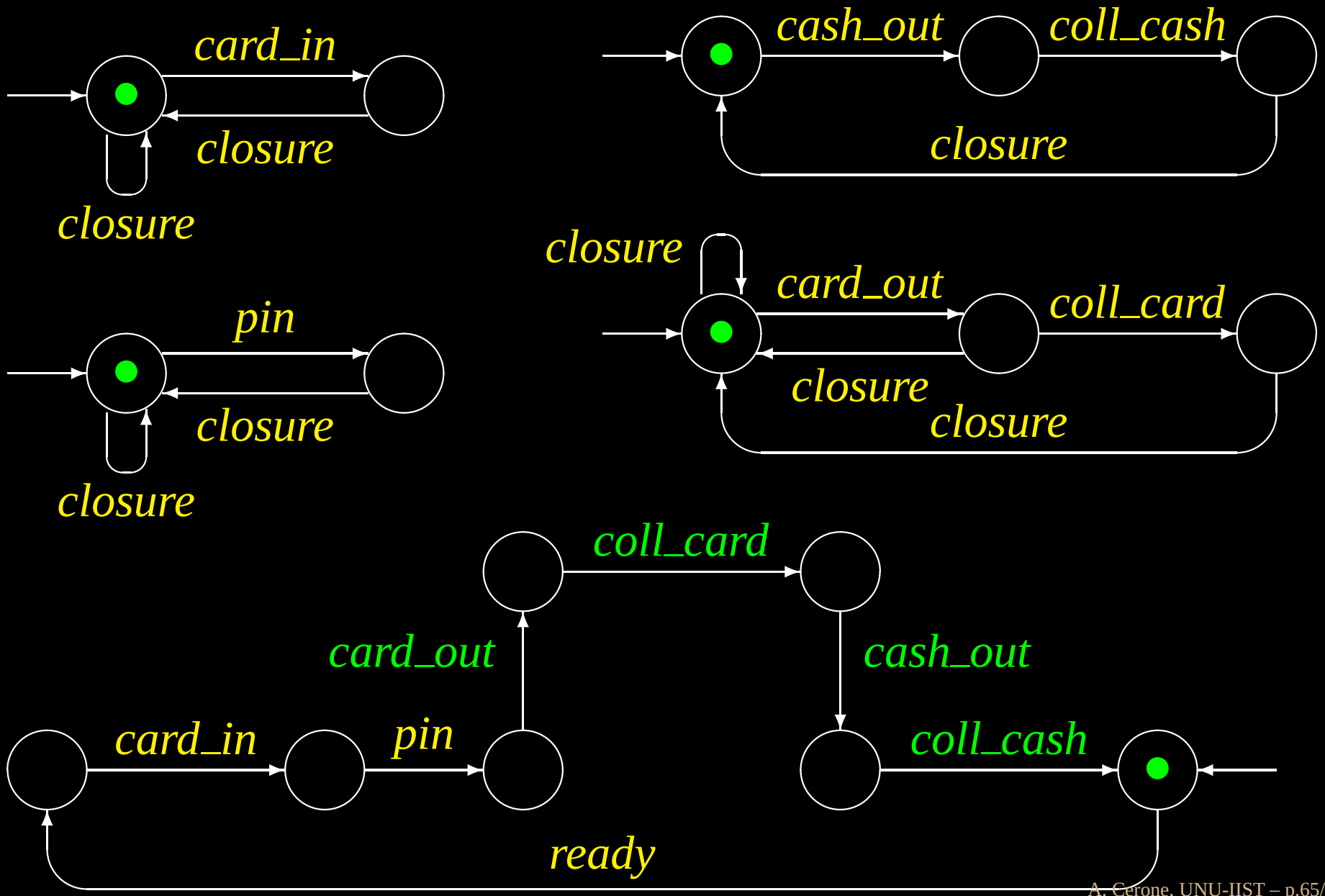
ATM: Post-completion Error



ATM: Correct Machine



ATM: Correct Machine



Closure: Exercise

How do you define the closure when you have more than one goal?

Model actions and closure for an ATM that allows to choose between

- cash withdrawal, and
- statements printing

References

[Lachman et al. 79]

R. Lachman, J. L. Lachman, E. C. Butterfield.
Cognitive Psychology and Information Processing.
Lawrence Erbaum Associates, 1979.

Cognitive Psychology Book
Describes the Computer Analogy.

[Card et al. 83]

Stuart K. Card, Thomas P. Moran and Allen Newell.
The Psychology of Human-Computer interaction.
Lawrence Erbaum Associates, 1983.

HCI General Book

Classical book that defines the early theoretical basis of HCI from an Information Processing perspective. Develops and describes the Model Human Processor in details.

End