Applications in human-computer interaction

Part 1: Qualitative Formal Analysis

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

Process Algebras

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

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

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

Starting CWB-NC

```
shell> cwb-nc csp

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

cwn-nc> load atm-machine.csp
Execution time (user,system,gc,real):(0.00
cwn-nc> help

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

CSP Notation

Communication Sequential Processes

Actions: a, b, c, ...

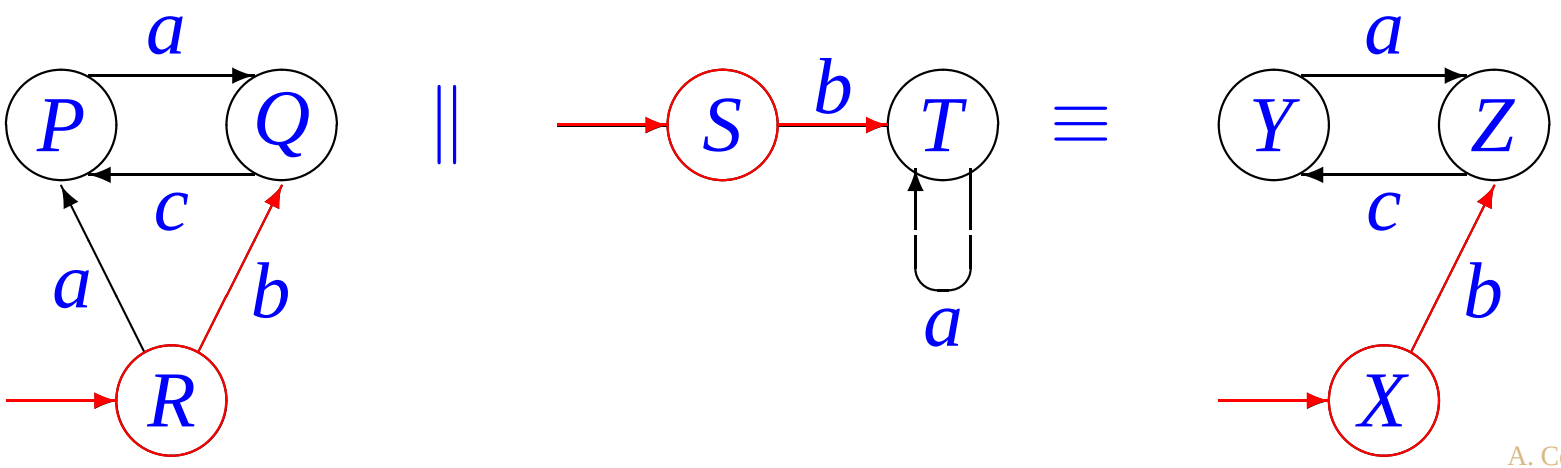
Processes: P, Q, R, ...

Basic Operators

Prefix: $\rightarrow$ $\quad P = a \rightarrow Q, \quad Q = c \rightarrow P$

Choice: $\Box$ $\quad R = (a \rightarrow P) \Box (b \rightarrow Q)$

Parallel: $\|$ $\quad R \parallel S, \quad S = b \rightarrow T, \quad T = a \rightarrow T$



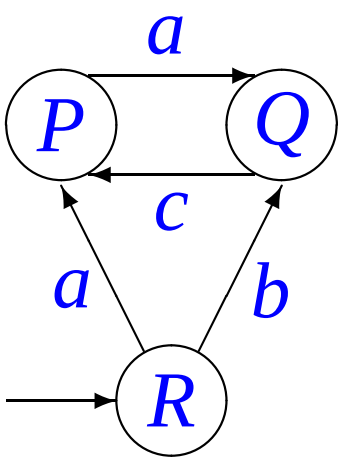
CWB-NC Syntax: R and S

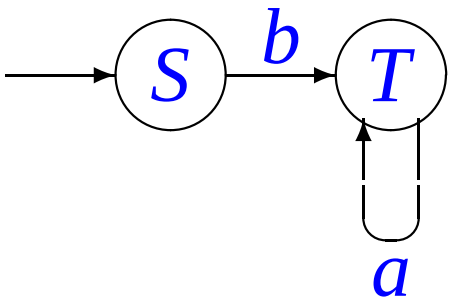
$P = a \rightarrow Q, \quad Q = c \rightarrow P, \quad R = (a \rightarrow P) \,[]\, (b \rightarrow Q)$

```
proc P = a -> Q
proc Q = c -> P
proc R = a -> P [] b -> Q
```

$S = b \rightarrow T, \quad T = a \rightarrow T$

```
proc S = b -> T
proc T = a -> T
```



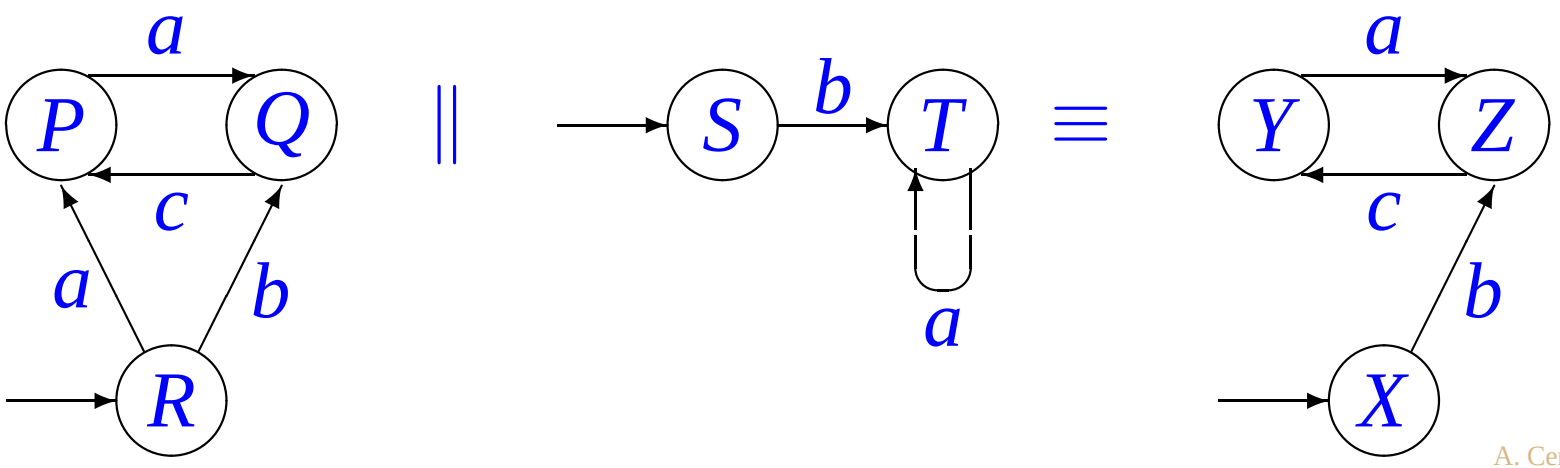


CSP: Synchronisation Set

$$X = R \parallel S \Longrightarrow X = R[|\{a,b\}|]S$$

$$sync\_ab = \{a,b\} \Longrightarrow X = R[|sync\_ab|]S$$

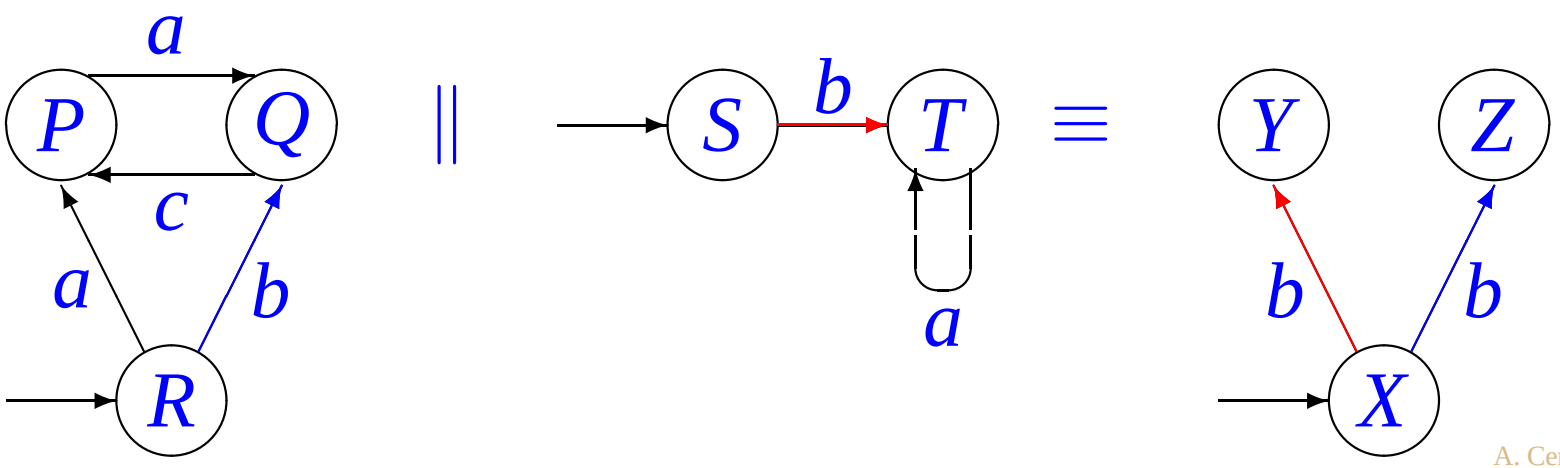
```
set sync-ab = { a,b }
proc X = R [| sync-ab |] S
```



CSP: Non-Determinism

$X = R \,[|\, \{a,c\} \,|]\, S$

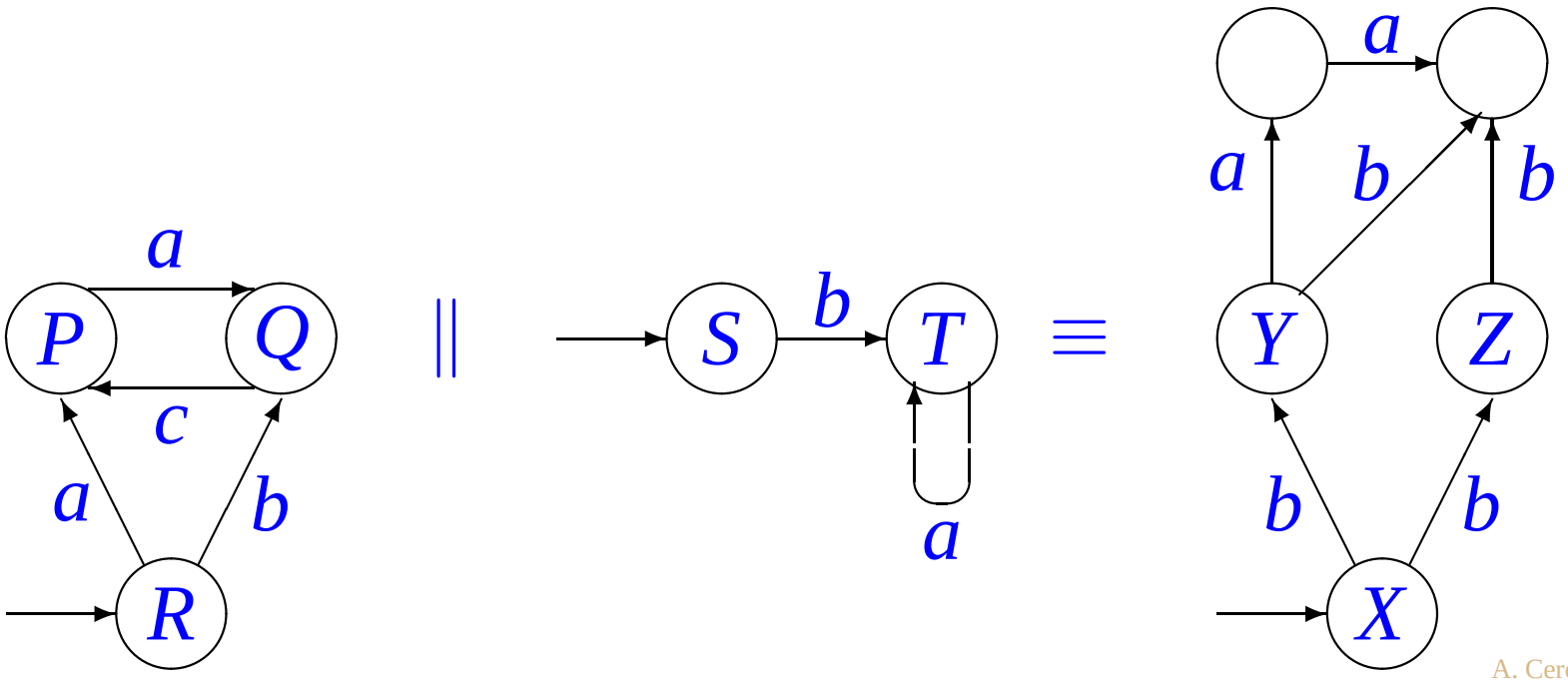
```
proc Xac = R [| {a,c} |] S
```



CSP: Deadlock

$X = R[|\ \{a, c\}\ |]S$

```
proc Xac = R [| {a,c} |] S
```



ATM Example

Example: ATM Machine

Informal Specification

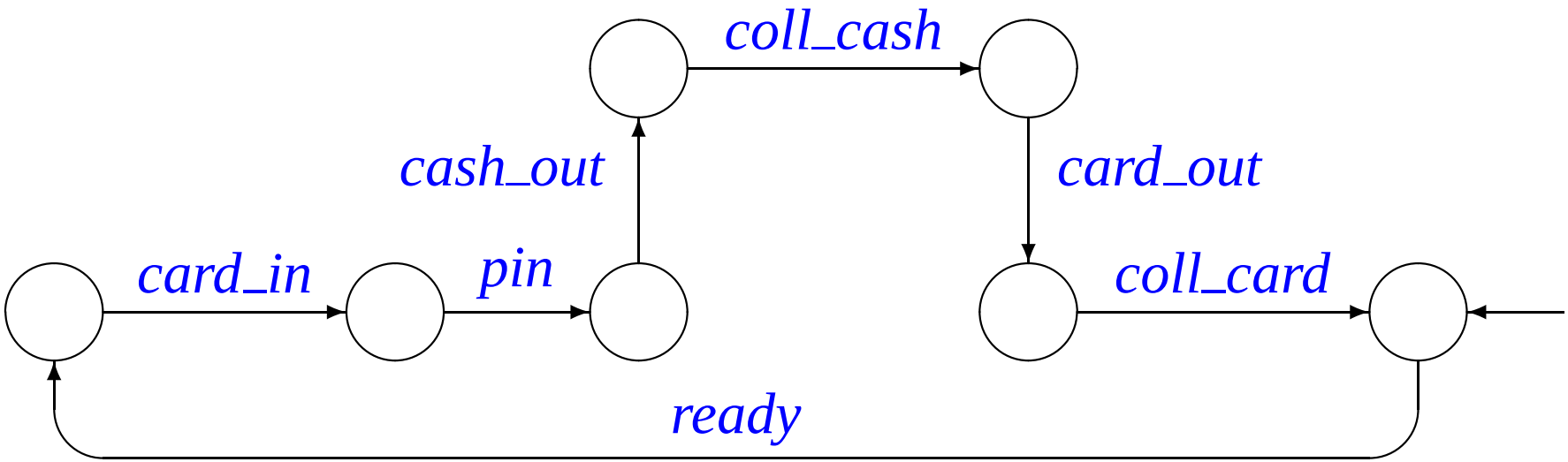
An ATM machine requires a user to

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

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



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> load atm-machine.csp
Execution time (user,system,gc,real):(0.00
cwn-nc> sim Machine
Machine
1:  -- ready --> card_in->cash_out->CashGi
cwb-nc-sim>1
card_in->cash_out->CashGiven
1:  -- card_in --> cash_out->CashGiven
cwb-nc-sim>quit
        Execution time (user,system,gc,rea
cwb-nc> quit
shell>
```

ATM Machine: Exercises

Modify the ATM

1. to allow a customer to choose between
   - cash withdrawal, and
   - statements printing
2. to take back card and/or cash if they are not collected by the customer within a given time
3. $1 + 2$

ATM Specification

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.

ATM Spec: "requires" part

cash delivered to the user

requires

bank card inserted
and
right pin for that card entered

cash_out requires *card_in*

((not *cash_out*) until *card_in*)
after the machine is *ready*

$$\forall \Box (ready \rightarrow ((\neg cash\_out) \,\mathcal{U}\, card\_in))$$

Missing Part

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.

ATM Spec: "allows" part

if

- bank card inserted
- right pin for that card entered

then

- cash delivered to the user
  before the machine is ready again
- card returned to the user
  before the machine is ready again
- the user has to collect the cash
  before the machine is ready again
- the user has to collect the card
  before the machine is ready again

ATM Spec: "allows" part

- if
  bank card inserted and later
  right pin for that card entered
  or
  right pin for that card entered and later
  bank card inserted
  then
  cash delivered to the user
  before the machine is ready again

$$\forall \Box((card\_in \wedge ((\neg ready)\,\mathcal{U}\, pin) \rightarrow ((\neg ready)\,\mathcal{U}\, cash\_out)) \vee (pin \wedge ((\neg cash\_out)\,\mathcal{U}\, card\_in) \rightarrow (\neg ready)\,\mathcal{U}\, cash\_out))$$

ATM Spec: "allows" part

- card returned to the user
  before the machine is ready again

  $\forall \Box (card\_in \rightarrow ((\neg ready) \,\mathcal{U}\, card\_out))$

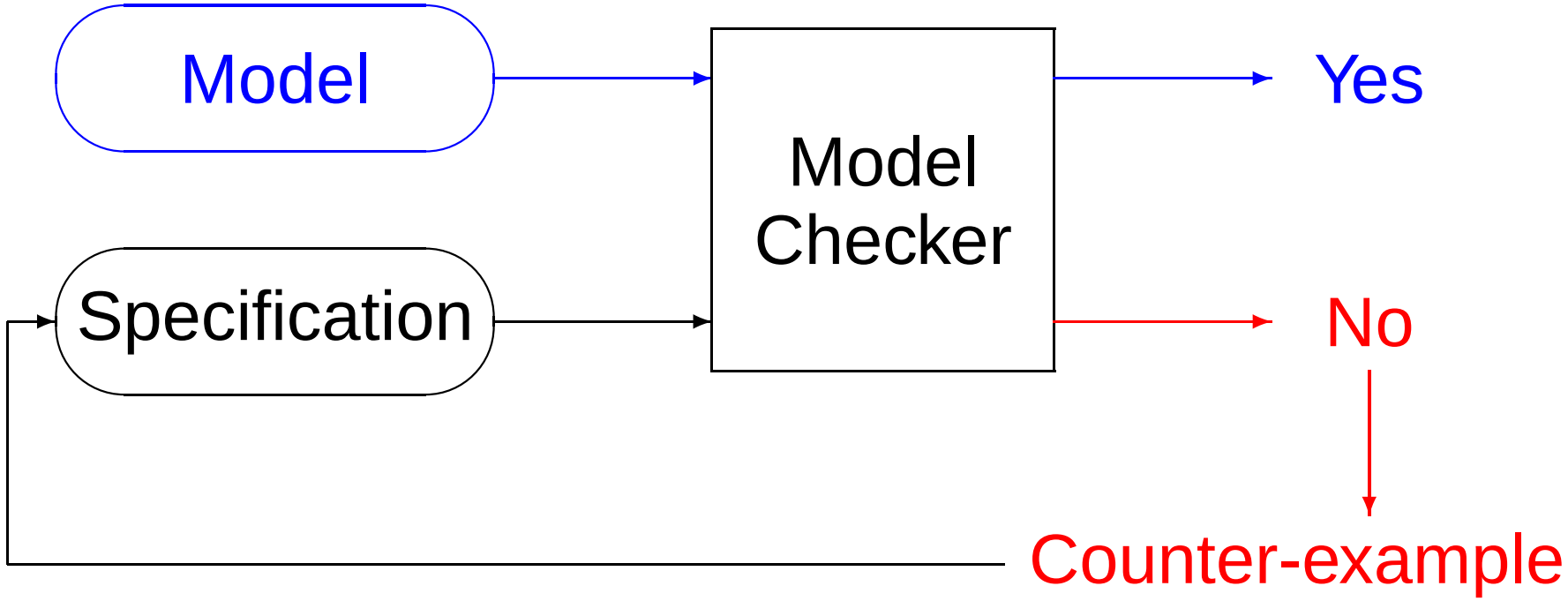
- the user has to collect the cash
  before the machine is ready again

  $\forall \Box (cash\_out \rightarrow ((\neg ready) \,\mathcal{U}\, coll\_cash))$

- the user has to collect the card
  before the machine is ready again

  $\forall \Box (card\_out \rightarrow ((\neg ready) \,\mathcal{U}\, coll\_card))$

Model-checking



HCI Concepts

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

$\Longrightarrow$ (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

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 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

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

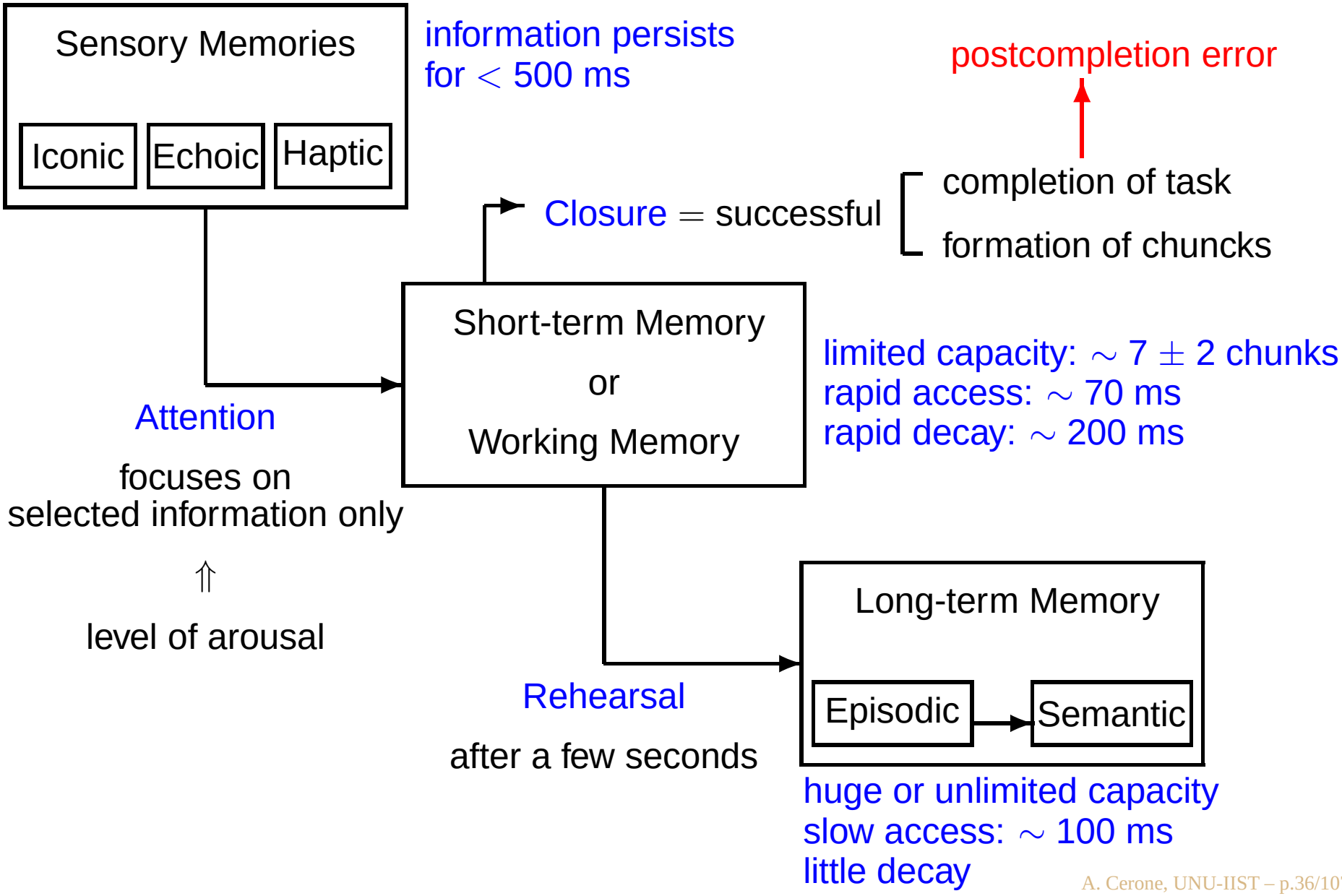
  independently of a specific computer/machine/interface
- about machine
  - expertise acquired through use/training
  - mental model

User Actions

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

involve the use of human memory

Human Memory



Modelling Human Behaviour

ATM Example Revisited

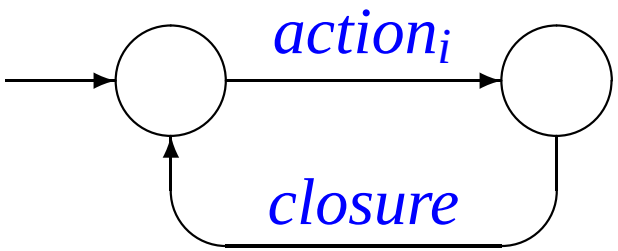
Cognitive Errors

Attention

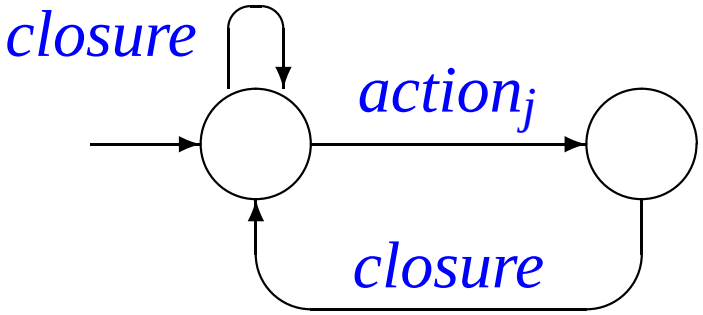
History of Formal HCI

Goals, Actions, Closure

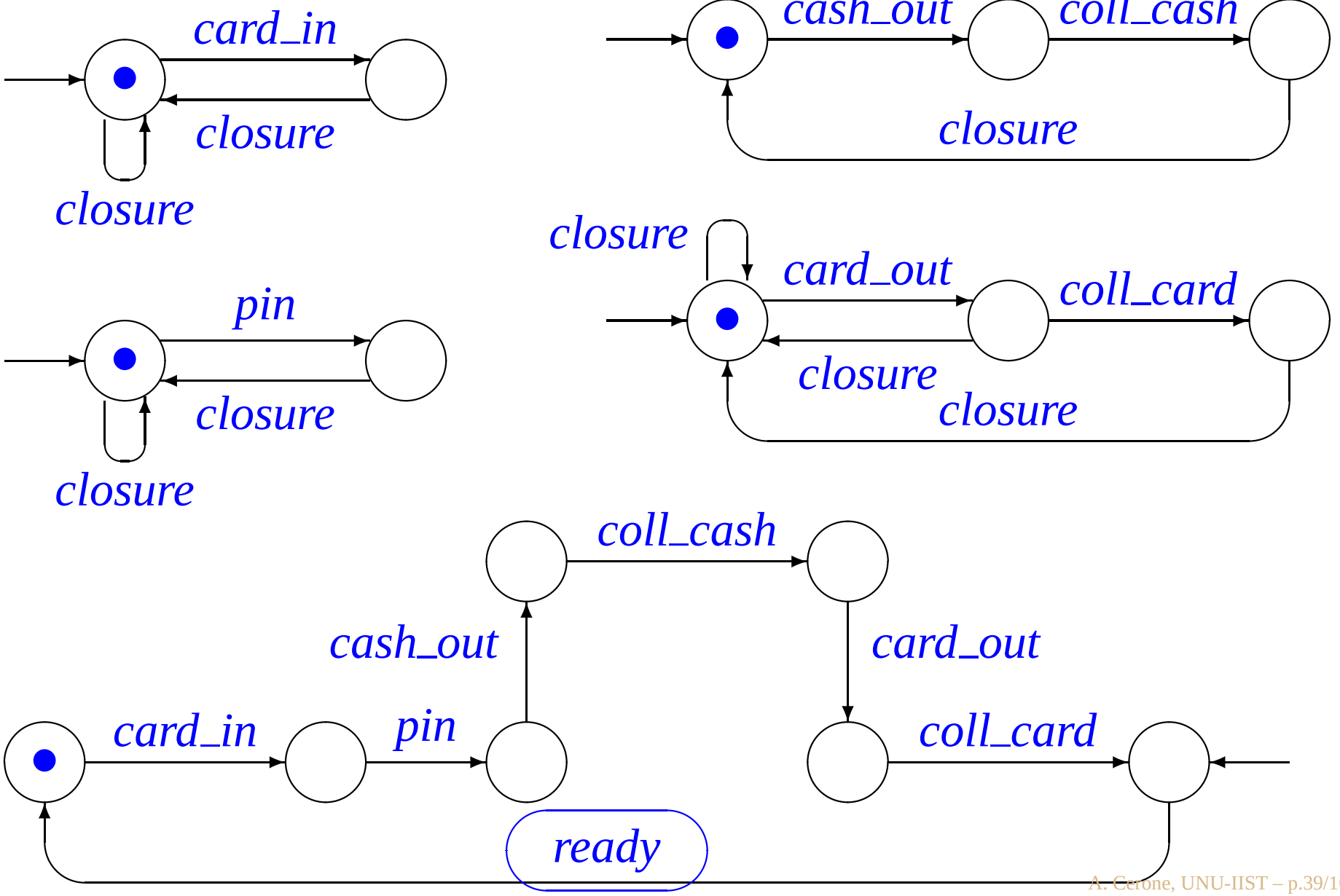
Goal action



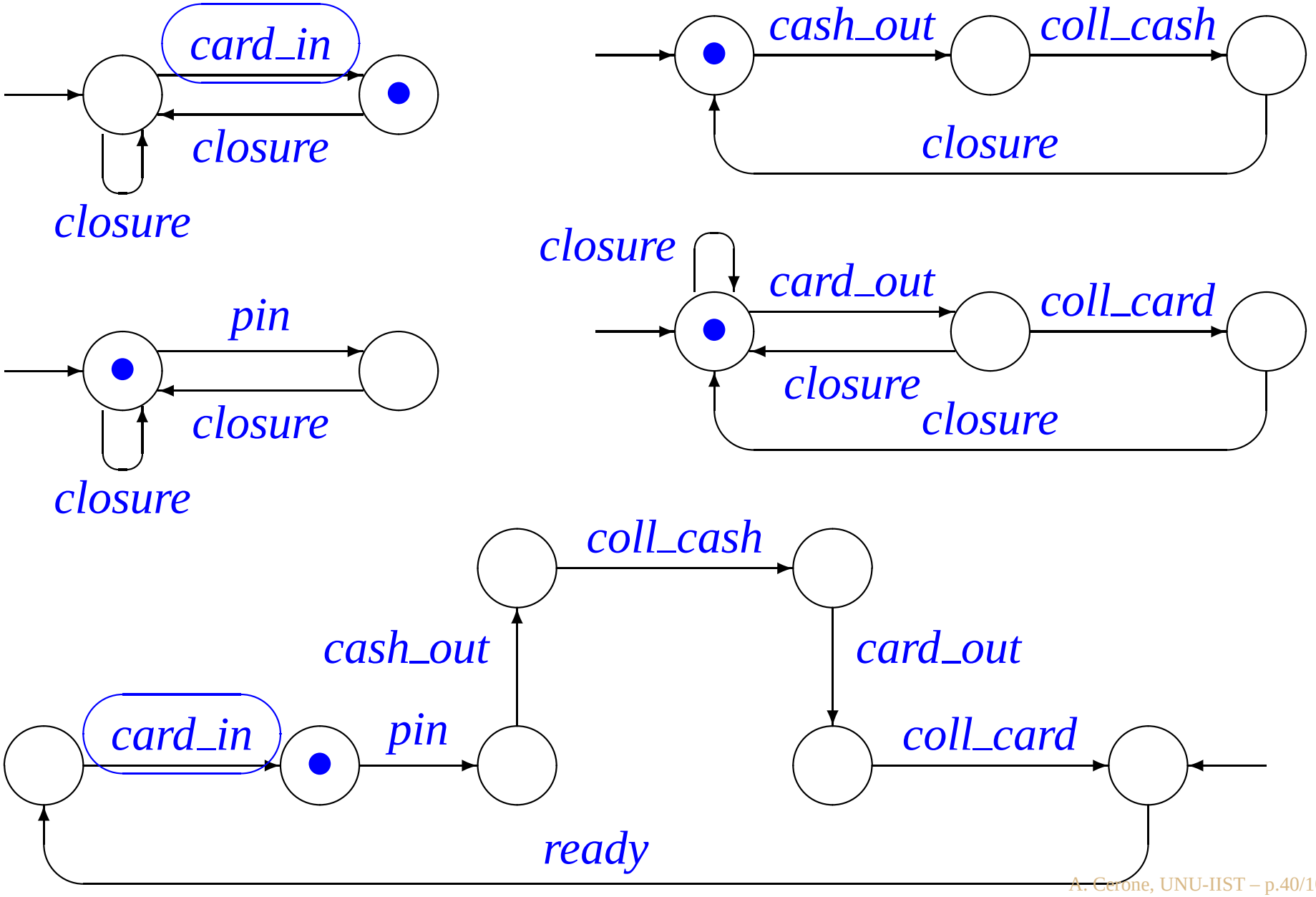
Other actions



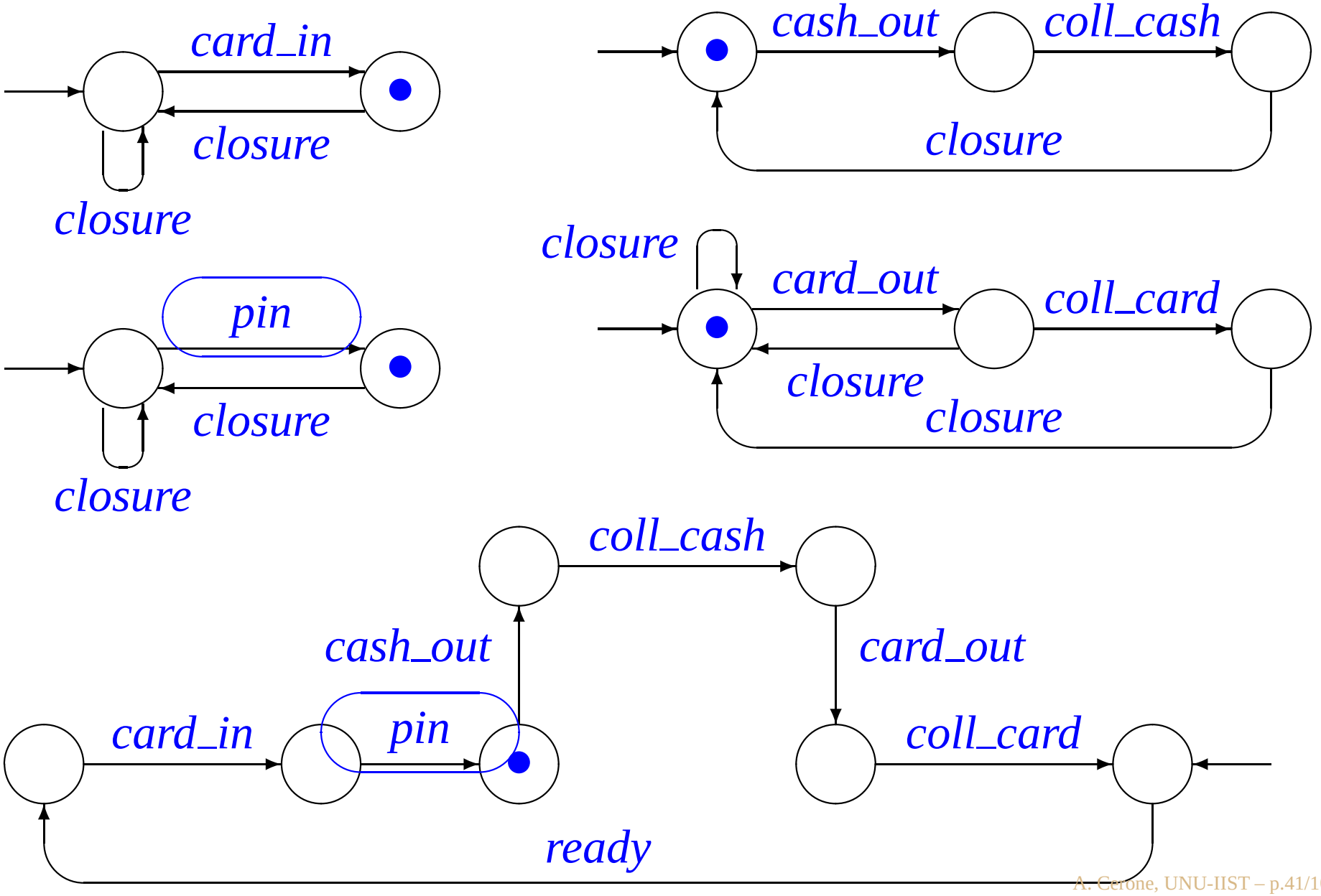
ATM: User


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

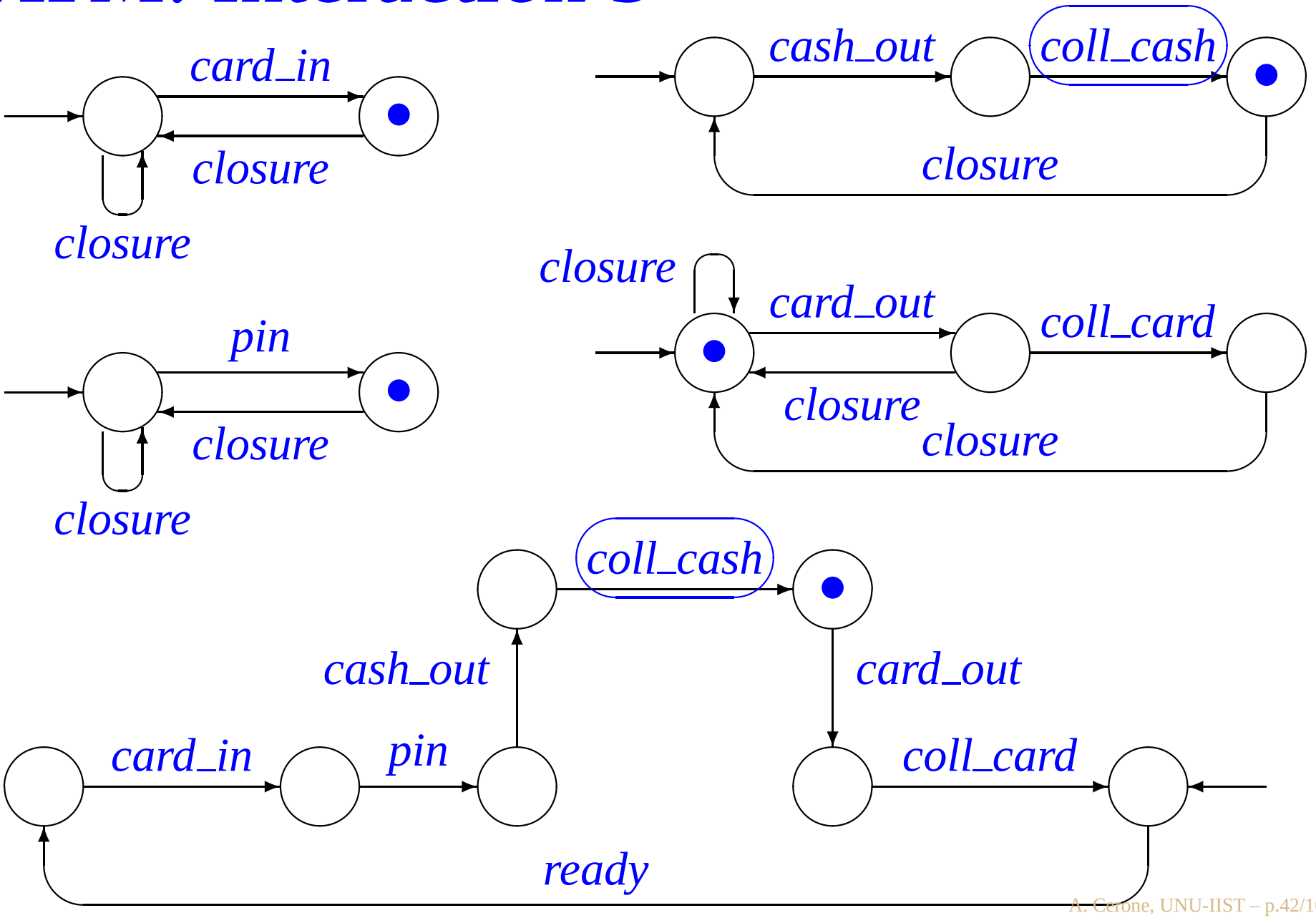
ATM: Interaction 1



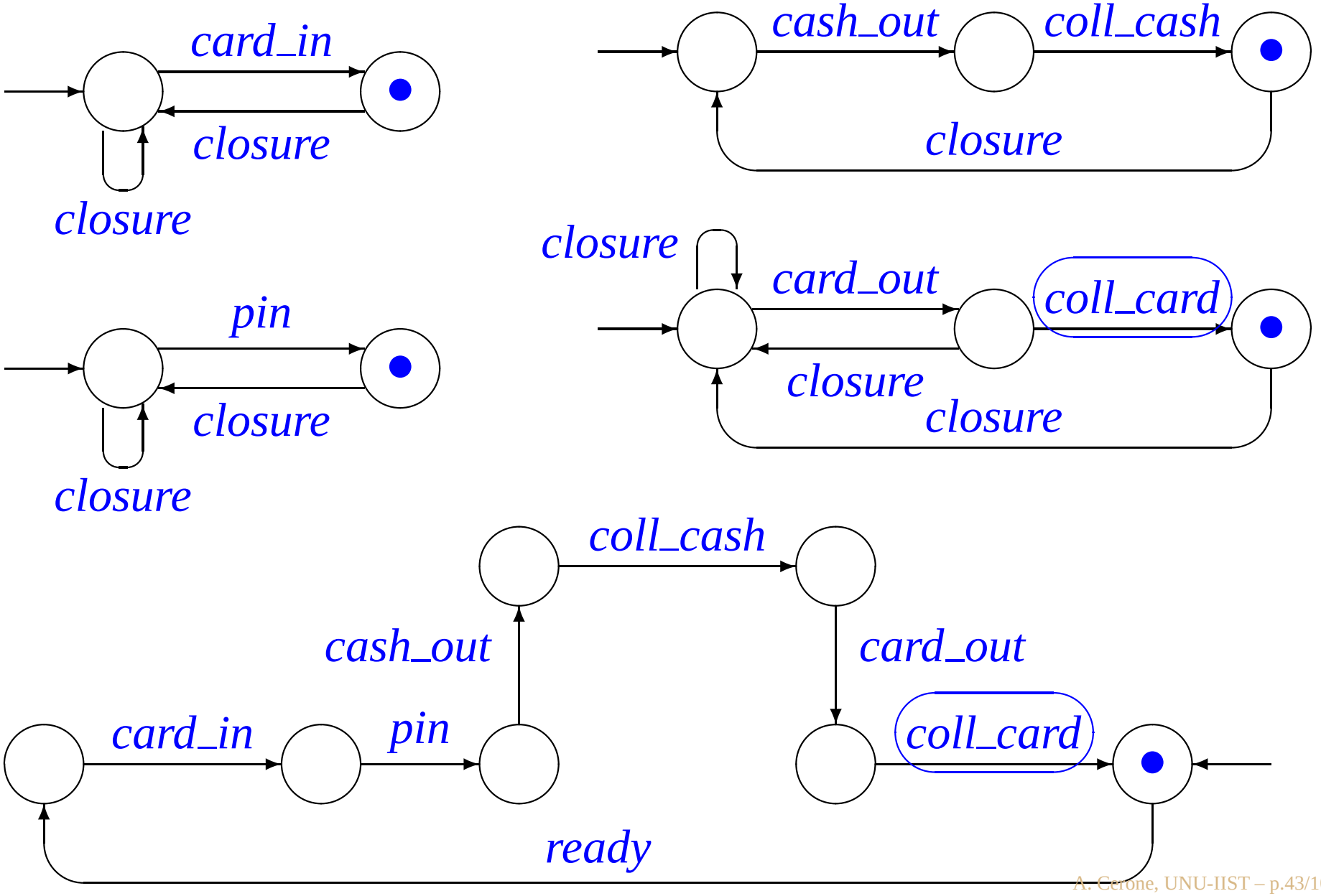
ATM: Interaction 2



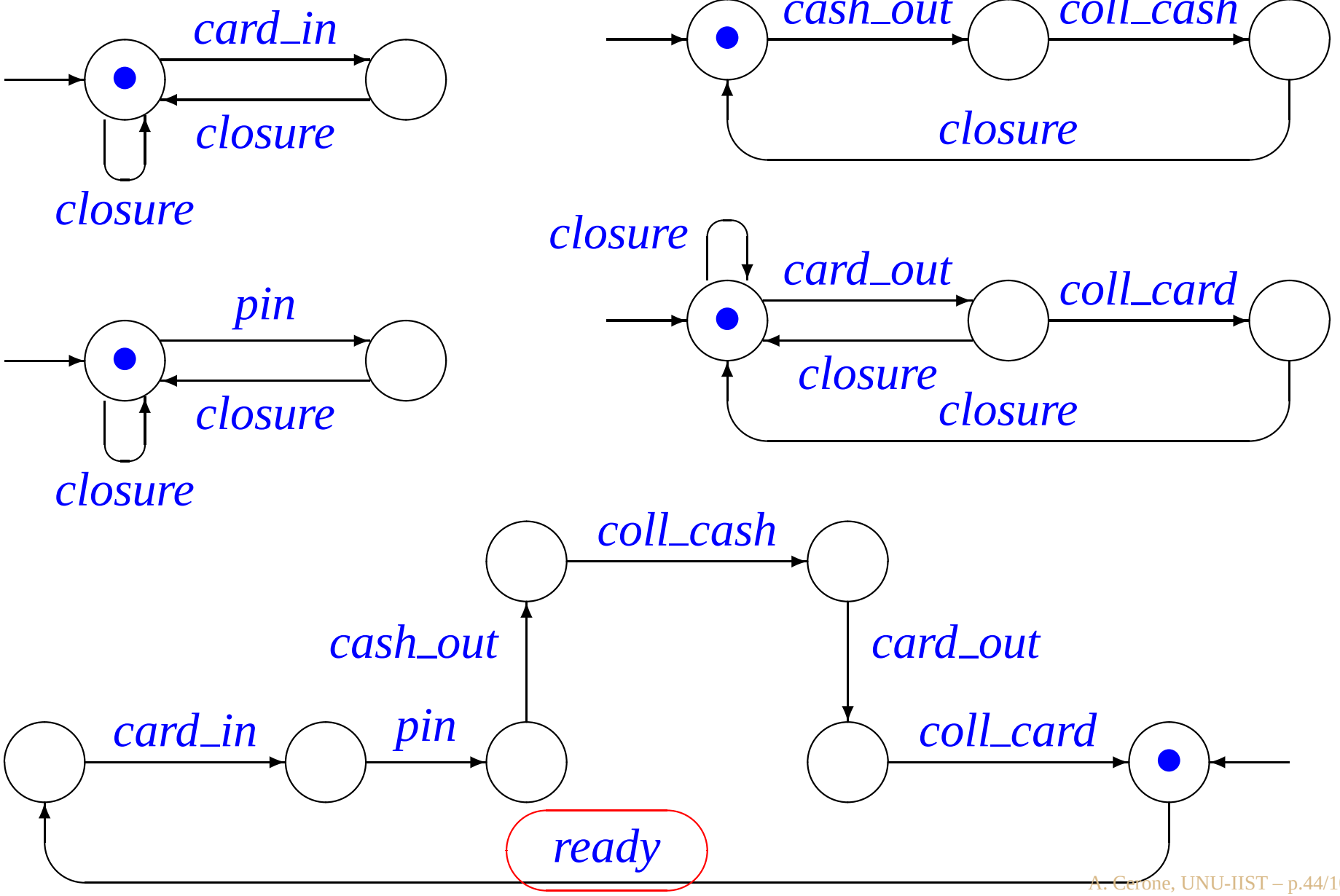
ATM: Interaction 3



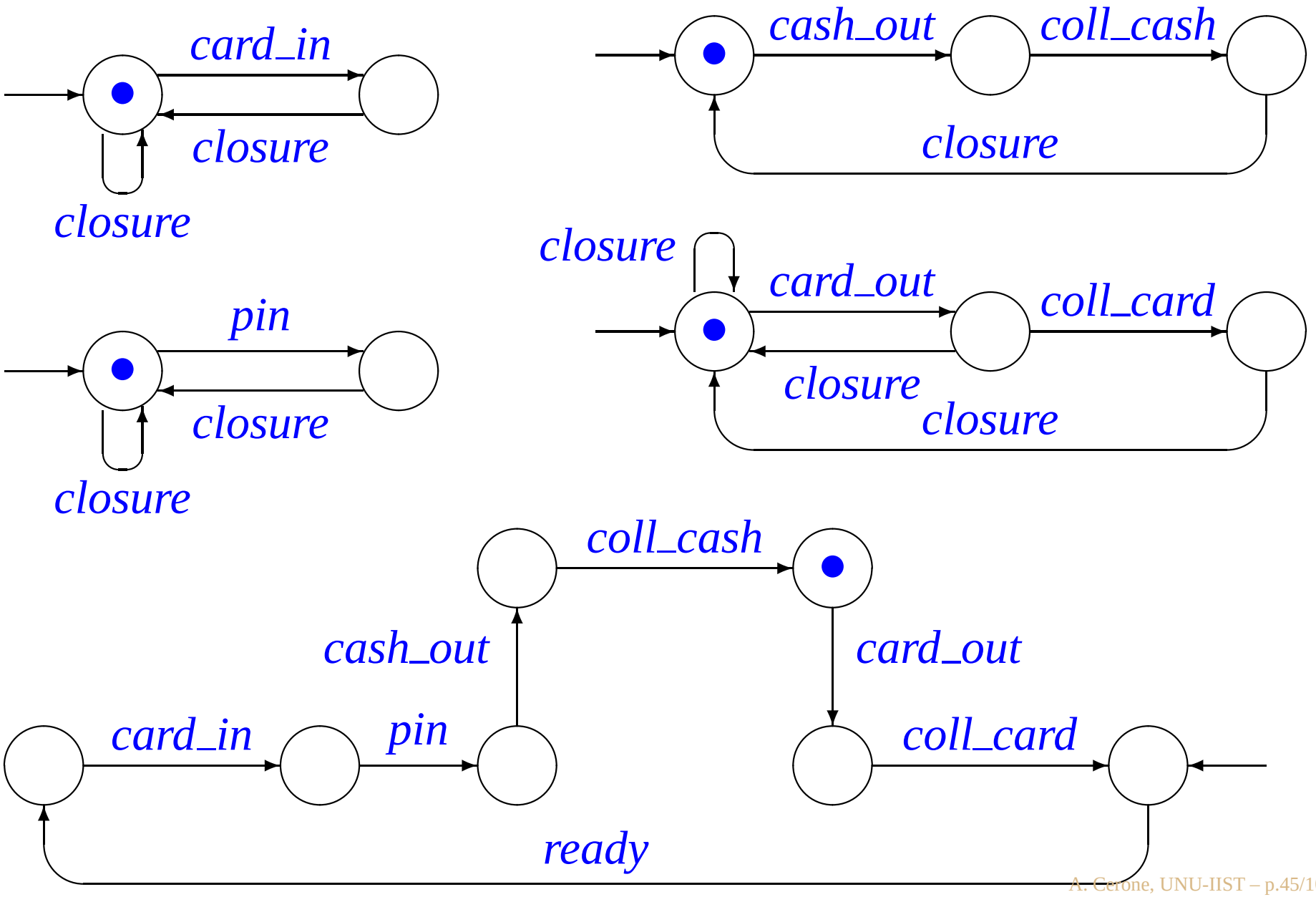
ATM: Interaction 4



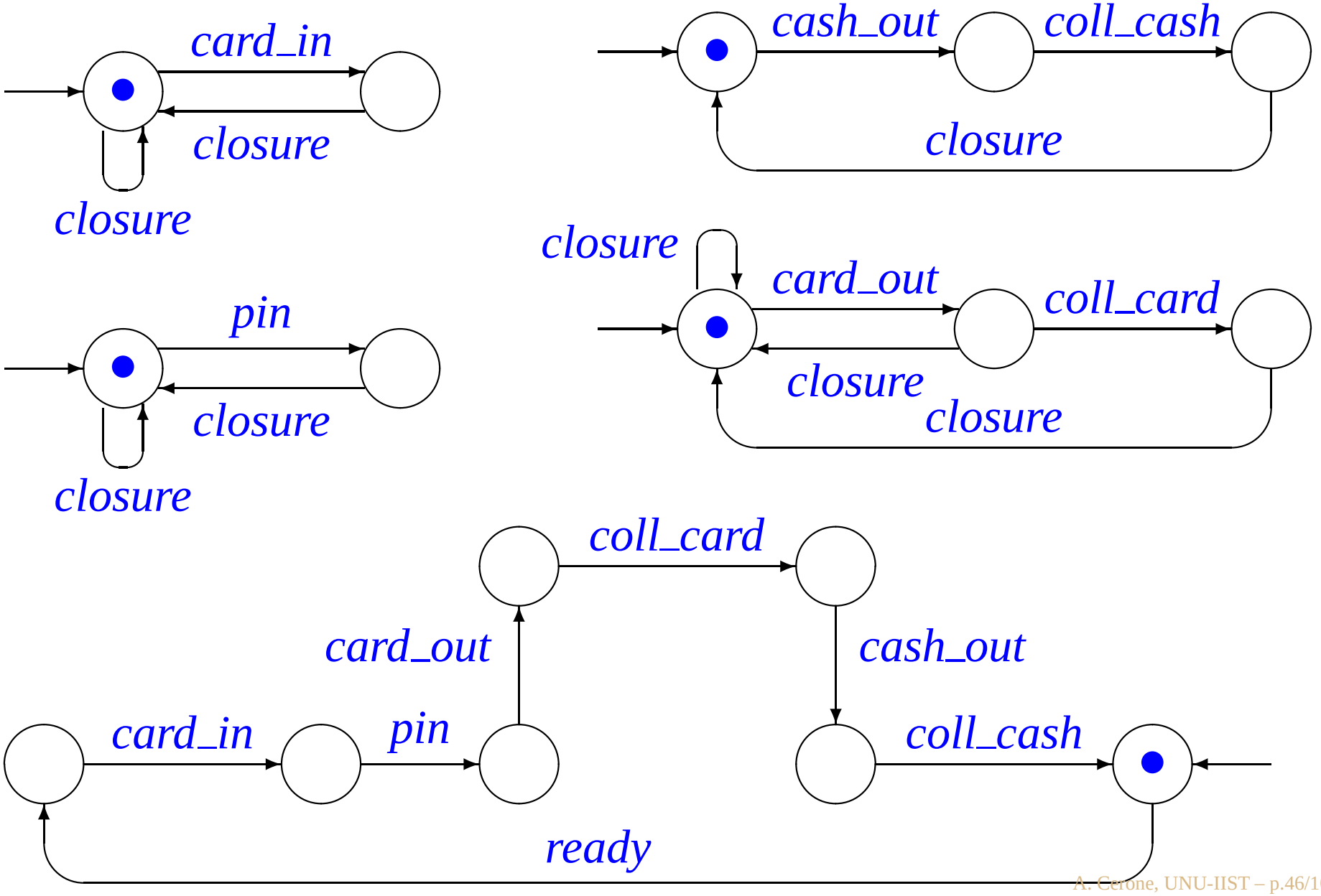
ATM: Closure


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: Correct Machine



Cognitive Errors

- Post-completion
- Impatiance
- Habituation $\Longrightarrow$ security violations

Thomas Anung Basuki, Antonio Cerone, Andreas Griesmayer and Rudolf Schlatte. Model-Checking User Behaviour Using Interacting Components. Formal Aspects of Computing, Vol. 21, No. 6, Springer, 2009.

- Limited Expertise $\Longrightarrow$ security violations

  Antonio Cerone and Norzima Elbegbayan. Model-checking Driven Design of Interactive Systems. ENTCS 183, Elsevier, pages 3–20, 2007.

Habituation: Exercises

1. Define a constraint that makes a user habituated to input the pin before inserting the card
2. Define a constraint that may eventually make a user habituated to input the pin before inserting the card

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

Attention

- selective attention
  (sensory memories $\Longrightarrow$ short-tem memory)
- attention versus automaticity
- Models of Attention
  - Norman and Shallice's Model
    - most responses: fairly automatic control
    - routine of responses
    - clash between routine activities
      $\Longrightarrow$ contention scheduling
    - routine activities inappropriate
      $\Longrightarrow$ attention activated by
      Supervisory Activating System (SAS)

Supervisory Activating System

SAS becomes active whenever the routine selection of operations becomes inappropriate

$\Longrightarrow$ whenever an individual encounters:

- required decision
- expectation failure

  assessed as
  - danger
  - novelty

  based on experience / mental model
- emotion
  - temptation, anger, ...

SAS activation in ATM

- required decision
  selections: kind of transaction, print balance
- danger
  card returned unexpectedly
- novelty
  keyboard on the screen,
  cash given at earlier stage
- temptation
  message: enter a draw if you withdraw ...
- anger
  message: no cash available

SAS activation in ATM (cont)

- required decision
  $\Longleftarrow$ choice operator
- danger
  $\Longleftarrow$ danger response = leave the interaction
- novelty
  $\Longleftarrow$ depends on the specific situation

Danger Response in ATM

```
proc Danger =
danger -> leave_int -> Danger
[] closure -> leave_int -> Danger
[] card_in Danger [] ...
```

The user will leave the interaction only in case of

- danger: user gives up achieving the goal
- closure: user has achieved the goal

We need to introduce a new action `leave_int` in the user model

Extended User Model — 1

Goal: collect cash

```
proc CollCashStart =
    start_int -> CollCashToDo
    cash_out -> CollCashStart

proc CollCashToDo =
    leave_int -> CollCashStart
 [] cash_out -> coll_cash
       -> CollCashDone

proc CollCashDone =
    closure -> leave_int
       -> CollCashStart
```

Extended User Model — 2

Non-goal Action: collect card

```
proc CollCardStart =
    start_int -> CollCardToDo
    card_out -> CollCardStart

proc CollCardToDo =
    leave_int -> CollCardStart
 [] closure -> CollCardToDo
 [] card_out -> coll_card
       -> CollCardDone

proc CollCardDone =
    leave-Int -> CollCardStart
 [] closure -> CollCardToDo
```

Extended User Model — 3

Non-goal Action: insert card

```
proc CardInStart =
    start_int -> CardToDo

proc CardToDo =
    leave_int -> CardInStart
 [] closure -> CardToDo
 [] card_in -> CardInDone

proc CardInDone =
    leave-Int -> CardInStart
 [] closure -> CardInToDo
```

Modelling SAS in ATM

- Routine Expectations $\Longrightarrow$ automaticity
  - expect card_out
  - expect cash_out
- Expectations Failure activates SAS
  - cash_out when card_out expected
  - card_out when cash_out expected
- Attention Response
  - assessment (danger or novelty)
  - action (`leave_int` or specific)

  based on experience / mental model

Routine Expectations in ATM

- expect cash_out before card_out

```
proc Expectations =
     pin -> expect_cash_out
        -> Expectations
  [] coll_cash -> expect_card_out
        -> Expectations
```

- expect card_out before cash_out

```
proc Expectations =
     pin -> expect_card_out
        -> Expectations
  [] coll_card -> expect_cash_out
        -> Expectations
```

Expectations Failure in ATM

```
proc SAS = start_int -> Activation
         [] card_out -> SAS
         [] csh_out -> SAS
```

Card Expectations Failure

```
proc SAS = start_int -> Activation
           [] card_out -> SAS
           [] csh_out -> SAS

proc Activation = expect_card_out ->
      ( card_out -> expect_met
          -> Activation
      [] cash_out -> cash_no_card
          -> Activation
      [] leave_int -> SAS )
  [] expect_cash_out -> ...
  [] leave_int -> SAS )
```

Cash Expectations Failure

```
proc SAS = start_int -> Activation
         [] card_out -> SAS
         [] csh_out -> SAS

proc Activation = expect_card_out ->
 ...
 [] expect_cash_out -> ...
     ( cash_out -> expect_met
        -> Activation
    [] card_out -> card_no_cash
        -> Activation
    [] leave_int -> SAS )
 [] leave_int -> SAS )
```

Interaction with SAS in ATM

```
proc Interaction_with_SAS =
  ( Interaction
      [| {start_int,card_out,
        cash_out,leave_int} |] SAS )
   [| {closure,leave_int,card_in,
     pin,coll_card,coll_cash} |]
  Danger
```

Failure Assessment in ATM

```
proc Assess =

 card_no_cash -> coll_card
   -> danger -> Assess      % danger

 cash_no_card-> coll_cash
   -> Assess                 % novelty

 expect_met ->
    (coll_cash -> Assess
 [] coll_card -> Assess)
```

based on task knowledge
and maybe experience / mental model

Attention Response in ATM

```
proc Attention_Response =
 ( Interaction_with_SAS
    [|{pin,expect_cash_out,
     coll_cash,expect_card_out}|]
  Expectations )
  [|{expect_met,
   card_no_cash,cash_no_card,
   coll_cash,coll_card,danger}|]
 Assess
```

MC Attention Response

- machine that delivers cash first
  - meets user expectation $\Longrightarrow$ MC: No
  - doesn't meet user expectation $\Longrightarrow$ MC: No
- machine that delivers card first
  - meets user expectation $\Longrightarrow$ MC: Yes
  - doesn't meet user expectation $\Longrightarrow$ MC: No

  Why?

  Because by receiving the card instead of the expected cash, the user believes the card has been rejected and is in danger of being confiscated if used again

History of Formal HCI

Safety Motivation

- 1980s: Human Reliability Assessment techniques [Svenson 1989, Kirwan 1990]
- 1990s: Formal Methods techniques for the analysis of
  - expected effective operator behaviour [Liskov and Wing 1994, Leveson 1990]
  - errors effectively performed by the operator [Johnson 1997]

But human behaviour is unpredictable

History of Formal HCI (cont.)

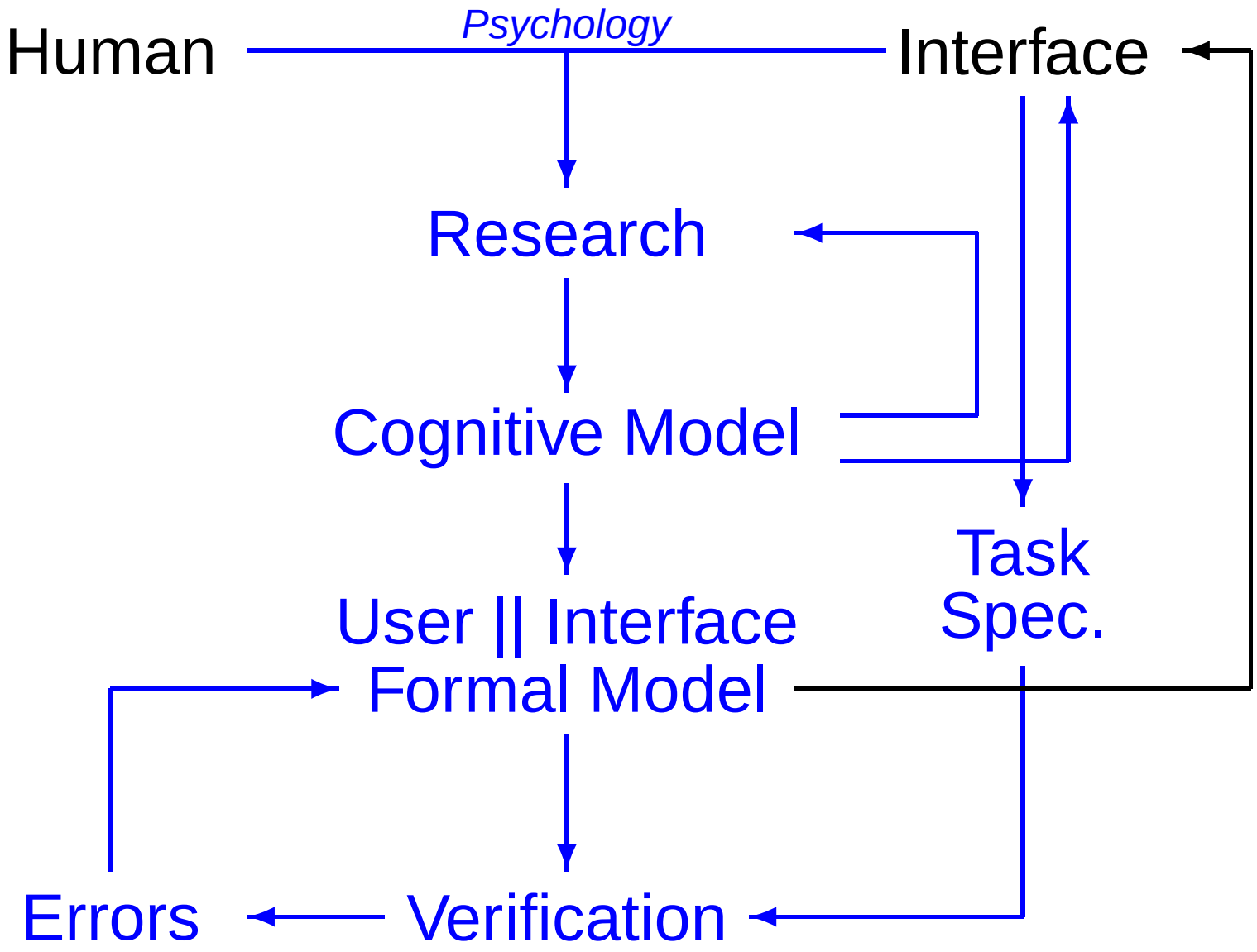
Unpredictable Behaviour

- end 1990s: Cognitively Plausible Behaviour [Butler et al. 1998, Butterworth et al. 2000, Rushby 2002, Curzon and Blandford 2004]

Security Motivation

- 2000s: Usability affects Security [Zurko 2005, Cerone and Curzon 2007]

Task Failure and ATC Example


Human
Psychology
Interface
Research
Cognitive Model
Task
Spec.
User || Interface
Formal Model
Errors
Verification

ATM Properties in LTL

Functional Correctness:

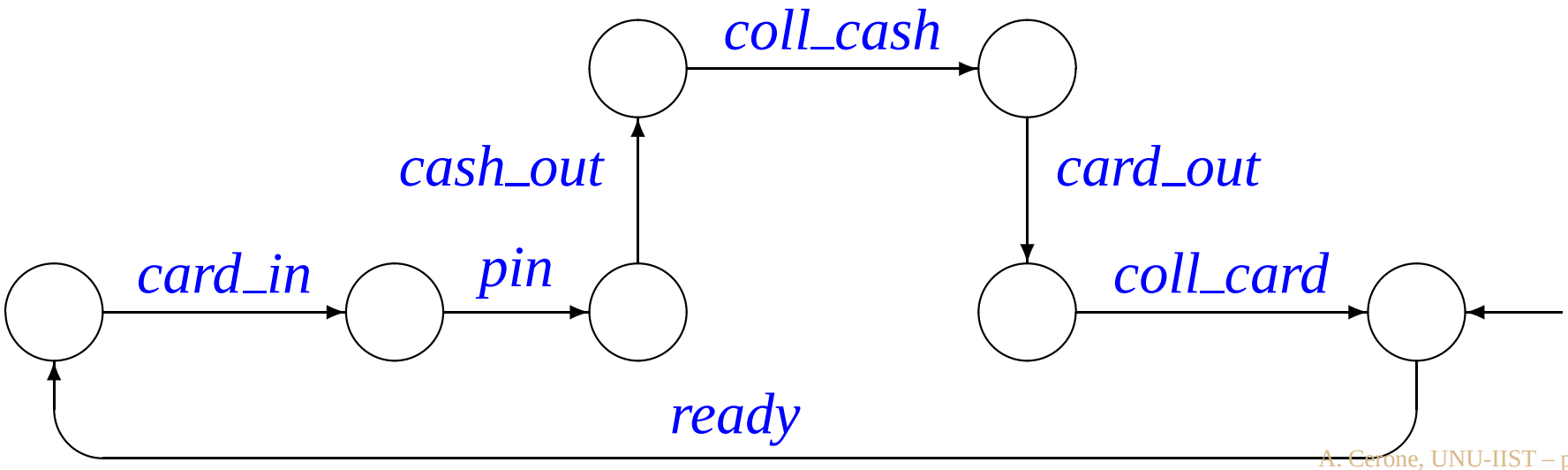
The ATM machine will eventually deliver cash

$$\Box(ready \rightarrow \Diamond cash\_out)$$

Safety:

The ATM machine will eventually return the card

$$\Box(ready \rightarrow \Diamond card\_out)$$

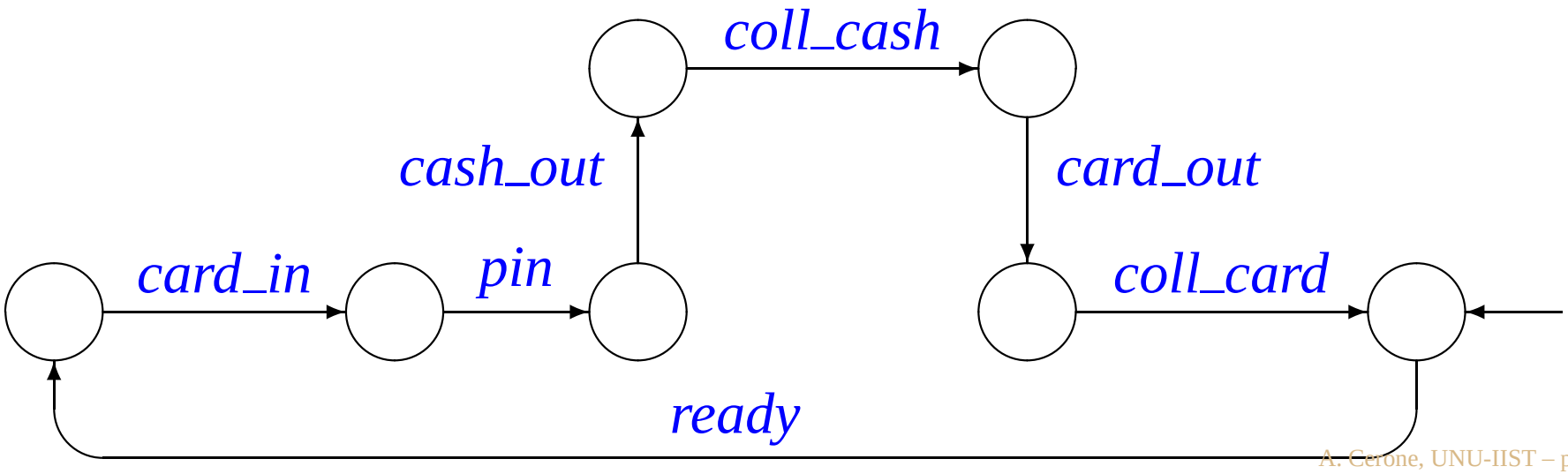


Example: ATM Machine

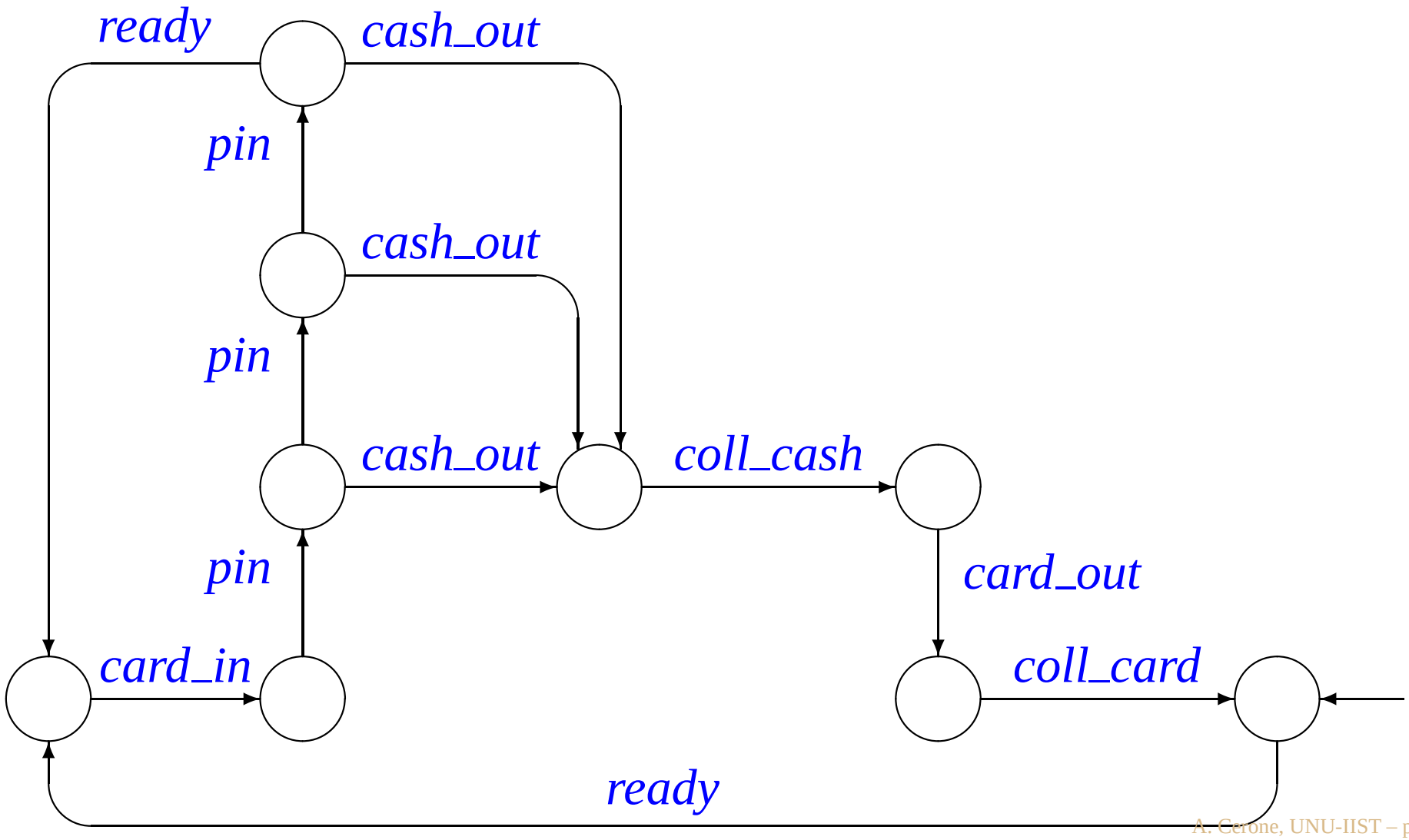
Goal: $\Box(ready \rightarrow \Diamond coll\_cash)$

Safety: $\Box(ready \rightarrow \Diamond coll\_card)$

Task: $\Box(ready \rightarrow ((\Diamond coll\_cash) \wedge (\Diamond coll\_card)))$



Refined ATM Machine



ATM: Task Failure

Goal: $\Box(ready \rightarrow \Diamond coll\_cash)$

Safety: $\Box(ready \rightarrow \Diamond coll\_card)$

Task: $\Box(ready \rightarrow ((\Diamond coll\_cash) \wedge (\Diamond coll\_card)))$

Task Failure:

$$(\Box\neg\, coll\_cash) \vee (\Box\neg\, coll\_card))$$

Task Failure Decomposition

Top-level Task Failure:

$(\Box\neg\, coll\_cash) \vee (\Box\neg\, coll\_card))$

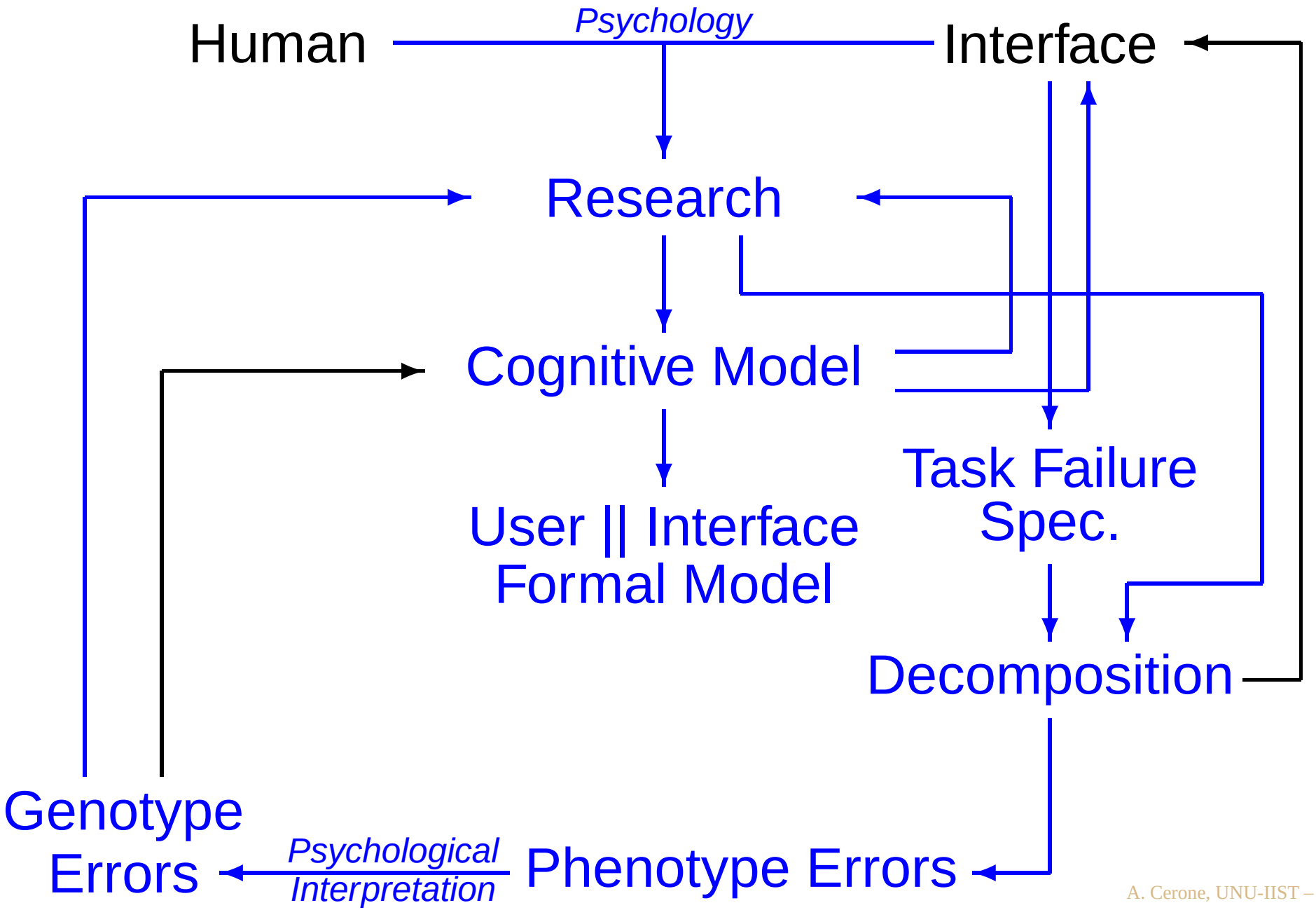
1. input wrong pin three times in a row
   $\Longrightarrow$ card confiscated and cash not collected
   $(\Box\neg\, coll\_cash) \wedge (\Box\neg\, coll\_card))$
2. collect cash but not card
   $(\Diamond coll\_cash) \wedge (\Box\neg\, coll\_card))$
3. collect card but not cash
   $(\Diamond coll\_card) \wedge (\Box\neg\, coll\_cash))$

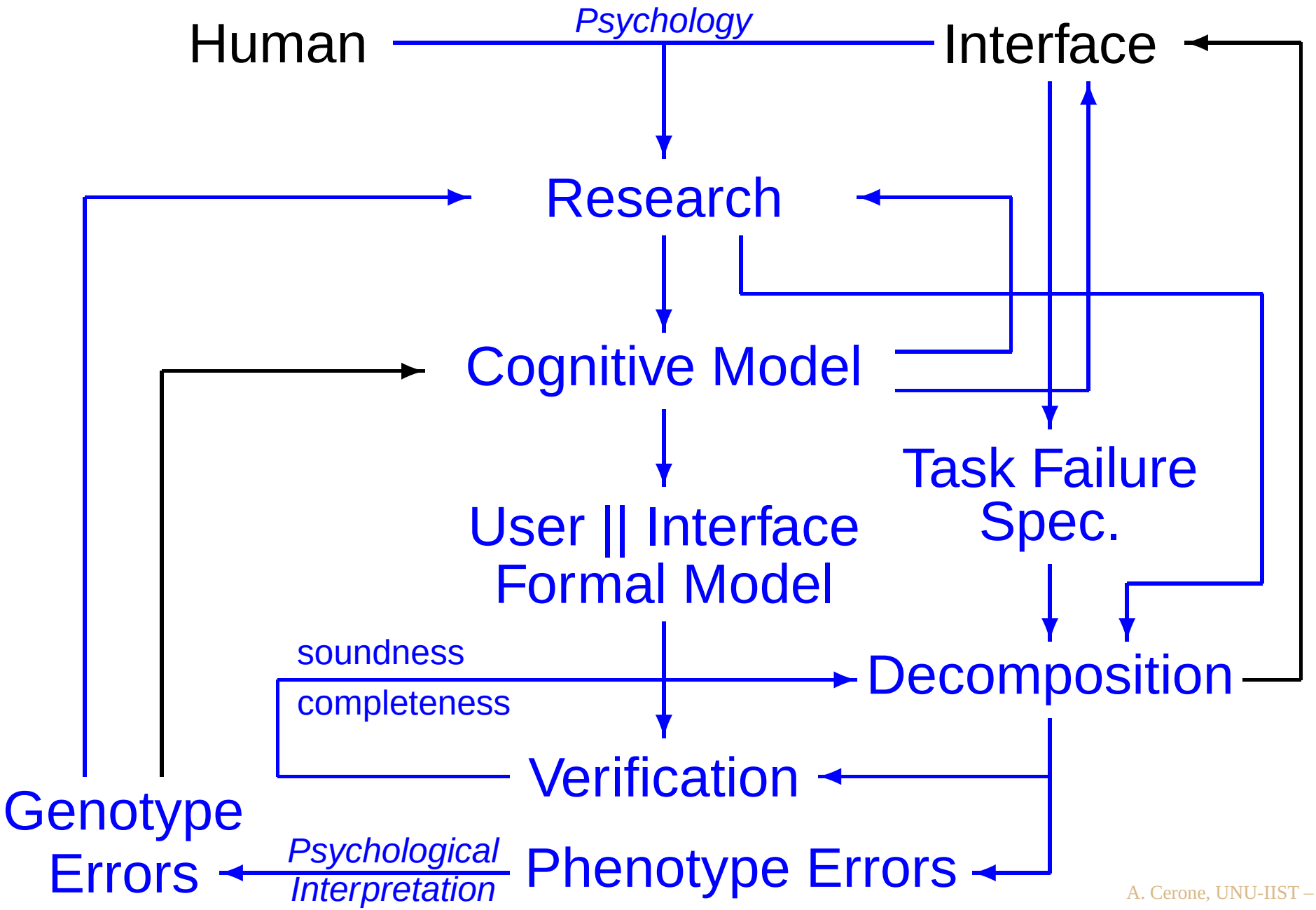
TF Psyc. Interpretation

Top-level Task Failure:

either card or cash is not collected

1. input wrong pin three times in a row
   $\Longrightarrow$ card confiscated and cash not collected
   $\Longleftarrow$ pin forgotten
2. collect cash but not card
   $\Longleftarrow$ forget to collect card due to postcompletion error
3. collect card but not cash
   $\Longleftarrow$ forget to collect cash


Human
Psychology
Interface
Research
Cognitive Model
User || Interface
Formal Model
Task Failure
Spec.
Decomposition
Genotype
Errors
Psychological
Interpretation
Phenotype Errors


Human
Psychology
Interface
Research
Cognitive Model
Task Failure
Spec.
User || Interface
Formal Model
soundness
completeness
Decomposition
Verification
Genotype
Errors
Psychological
Interpretation
Phenotype Errors

Operator Choice Model

Scanning: The operator searches the interface for a certain property.

Identification: The operator identifies part of the interface that may represent the property.

Classification: The operator

- assesses whether the property is in need of further interest;
- if so, gives some form of priority to the property.

Decision on how to resolve the situation.

Action to be performed as a series of interaction with the interface.

OCM for Nuclear Plant

Scanning: The operator scans among each of the individual reactor readouts on the interface searching for any anomalies.

Identification: The operator identifies a particular readout.

Classification: The operator

- assesses whether the identified readout describes a normal or abnormal operation of the plant;
- if abnormal, gives a priority to the operation according to its urgency to be resolved.

Decision on how to resolve the abnormal situation.

Action to be performed as a series of interaction with the interface and with internal and/or external authorities.

OCM for Air Traffic Control

Scanning: The operator scans among each pair of aircraft searching for a pair that may violate separation.

Identification: The operator identifies a pair of aircraft.

Classification: The operator

- assesses whether the identified pair of aircraft will eventually violate separation (in conflict) or not (not in conflict);
- if so, gives a priority to the conflict according to its urgency to be resolved.

Decision on how to resolve the conflict.

Action to be performed as a series of interaction with the interface.

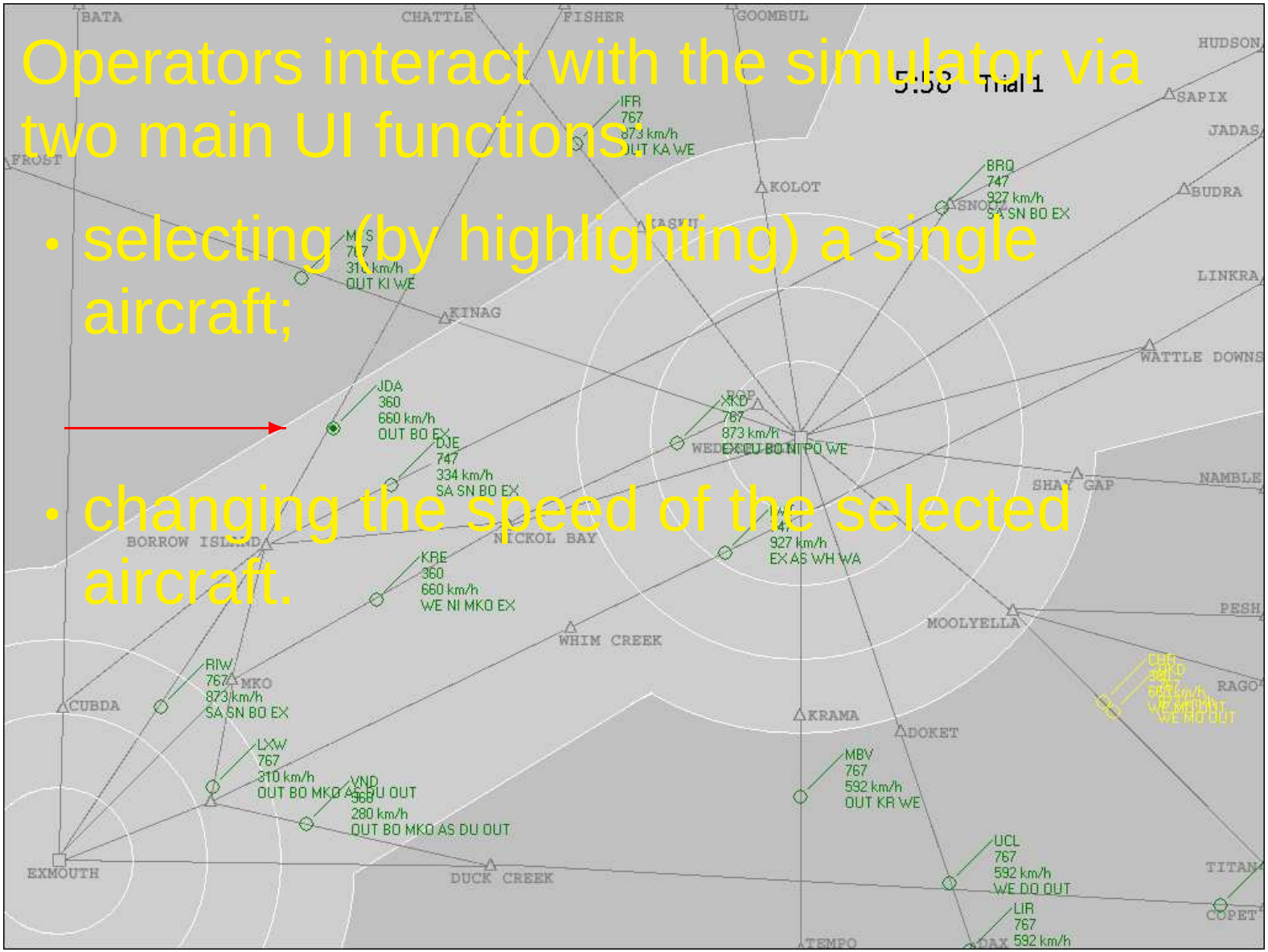
Air Traffic Control (ATC)

- Aircraft fly along straight-line segments — called *flight paths* — between *waypoints* within a fixed sector of airspace.
- Aircraft *horizontal separation* must be at least 5 miles.
- A *pair* of aircraft *violate separation* when the horizontal distance between them is less than 5 miles (*separation violation*).
- A *pair* of aircraft is in *conflict* when their pathways are such that the two aircraft will eventually violate separation.

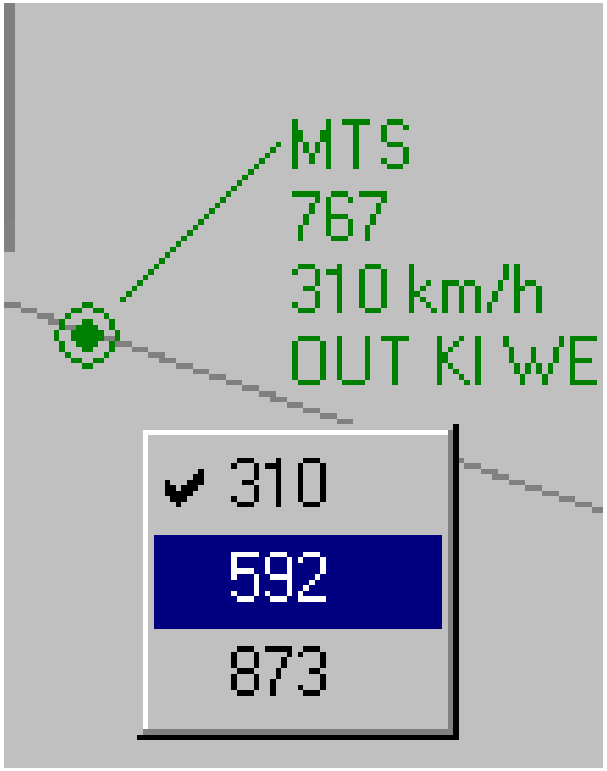
ATC Simulator

- The ATC operator's task involves monitoring the movement of aircraft on a screen, looking for pair of aircraft that *may violate separation*.
- When such a conflict is detected, the operator uses a mouse to select one of the aircraft and change its speed using a pulldown menu.
- The goal of the task is to resolve all conflicts before they violate separation, while not introducing any new conflict.
- We have a task failure when separation is violated.

ATC Simulator Screenshot



Speed Menu



Open the menu by clicking the right button.

The menu appears at the position of the cursor.

Selected the speed by left clicking on the desired menu entry.

Operator Erros

- slip: inadvertedly select a wrong or the current speed
  $\Longleftarrow$ selction task closure (cognitive problem)
- mistaken identity: change the spead of an aircraft different from the intended one
  $\Longleftarrow$ the menu appears at the position of the cursor (usability problem)
- mis-classification, mis-prioritization, conflict generation

The operator can recover from these errors without causing separation violation (task failure)

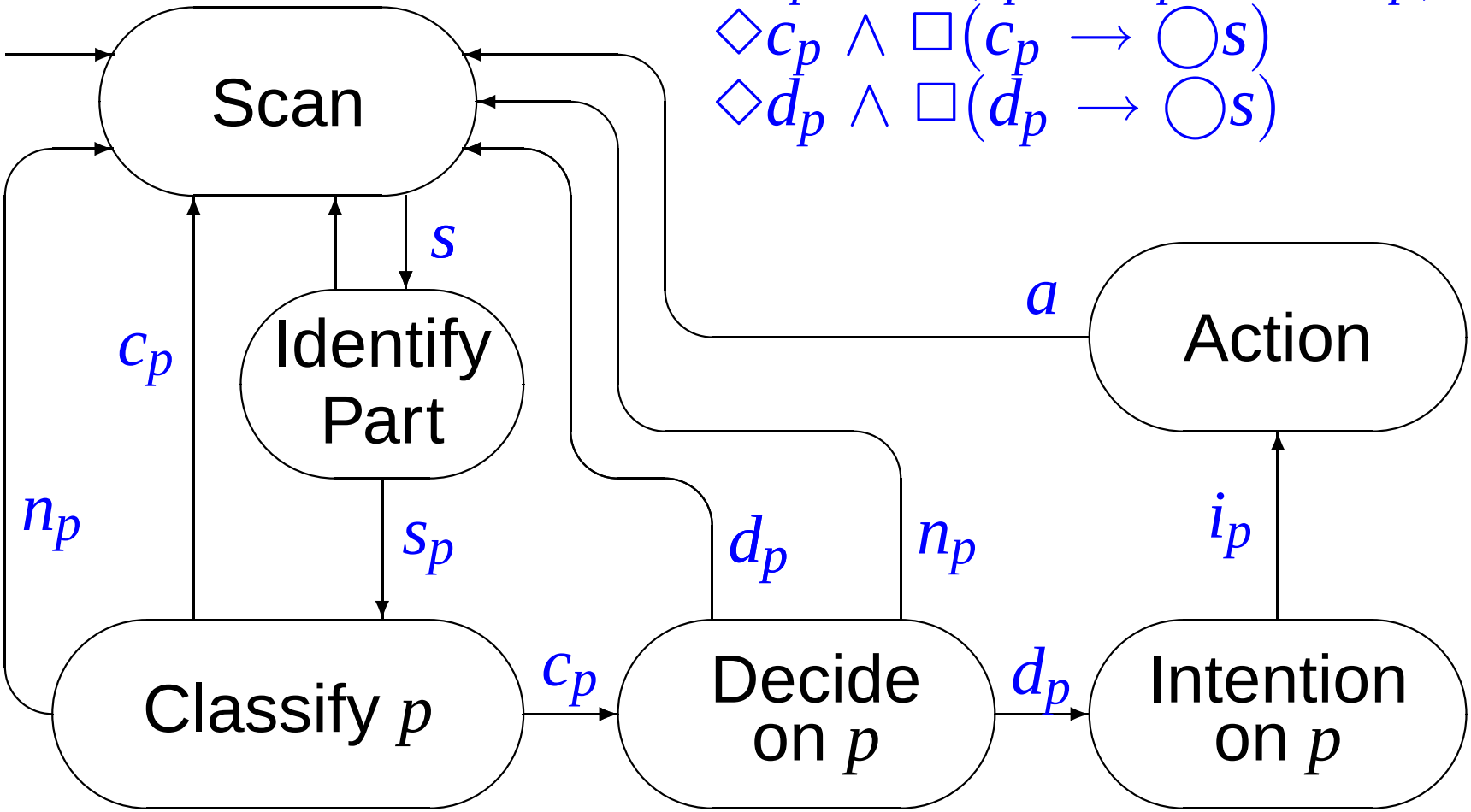
Task Failure Decomposition

$\Box\neg\, i_p$ is decomposed as $\Box\neg\, s_p$

$\Diamond s_p \wedge \Box(s_p \vee c_p \rightarrow \bigcirc n_p)$

$\Diamond c_p \wedge \Box(c_p \rightarrow \bigcirc s)$

$\Diamond d_p \wedge \Box(d_p \rightarrow \bigcirc s)$



Single Mis-prioritisation

$c_p \rightarrow \bigcirc s$

(phenotype error)

Possible genotype errors are

- mis-calculation
- mis-storage
- mis-retrival

of the time planned for corrective actions
$\Longrightarrow$ possible recovery
(through new calculation at a next scan)

Persistent Mis-prioritisation

$\Diamond c_p \wedge \Box(c_p \rightarrow \bigcirc s)$

(phenotype error)

Possible genotype errors

- perception distorted $\Longrightarrow$ memory of result of previous mis-calculation keeps emerging

due to

- distraction
- similarity with observed non-conflicts
- high workload

ATC: References

Peter Lindsay and Simon Connelly. Modelling Erroneous Operator Behaviours for an Air-Traffic Control Task. AUIC 2002.

Antonio Cerone, Peter Lindsay and Simon Connelly. Formal Analysis of Human-computer Interaction using Model-checking. SEFM 2005.

Antonio Cerone, Simon Connelly and Peter Lindsay. Formal Analysis of Human Operator Behavioural Patterns in Interactive Surveillance Systems. Software and System Modeling 7(3), Springer, pages 273–286, 2008.

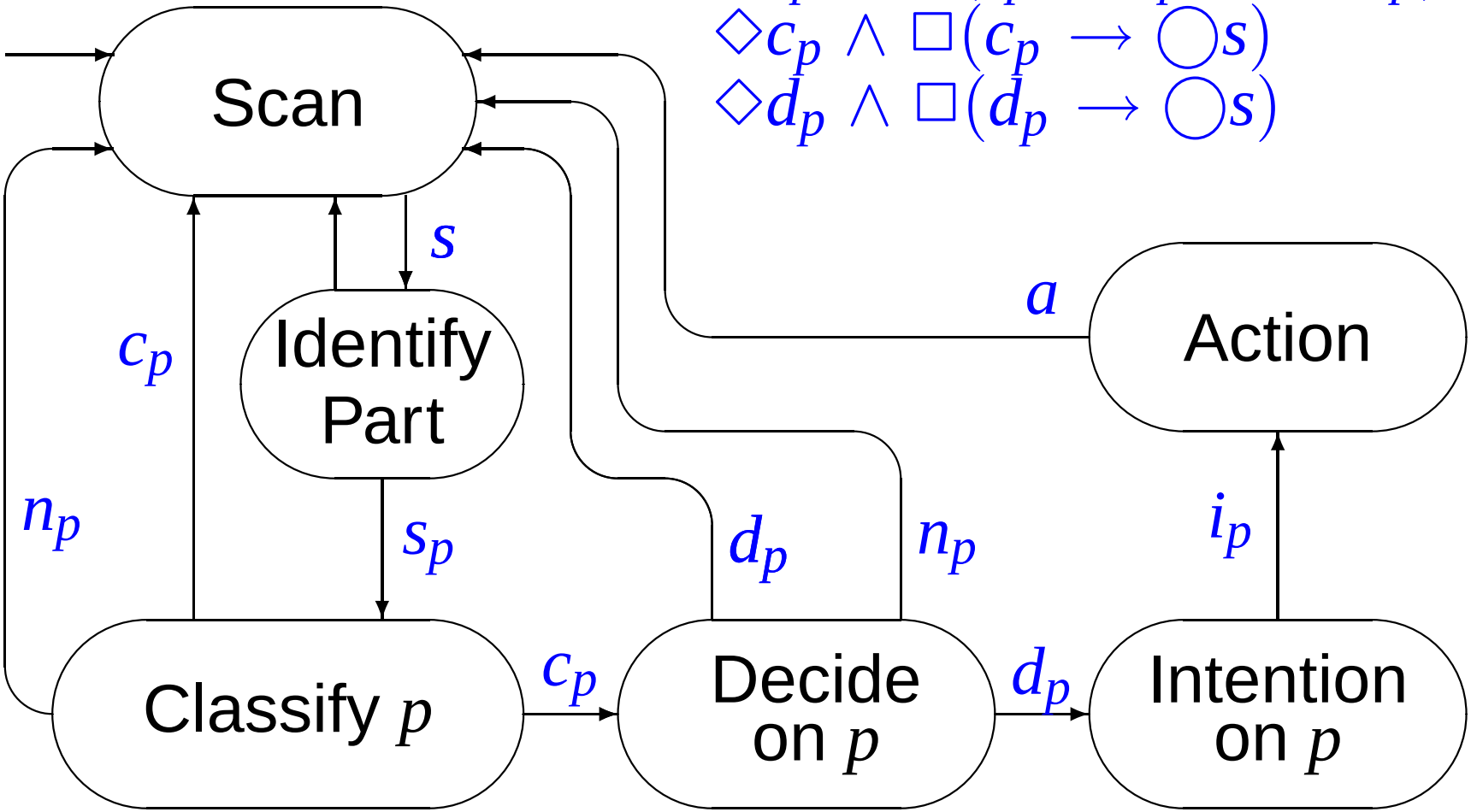
Task Failure Decomposition

$\Box\neg\, i_p$ is decomposed as $\Box\neg\, s_p$

$\Diamond s_p \wedge \Box(s_p \vee c_p \rightarrow \bigcirc n_p)$

$\Diamond c_p \wedge \Box(c_p \rightarrow \bigcirc s)$

$\Diamond d_p \wedge \Box(d_p \rightarrow \bigcirc s)$



Complete Decomposition

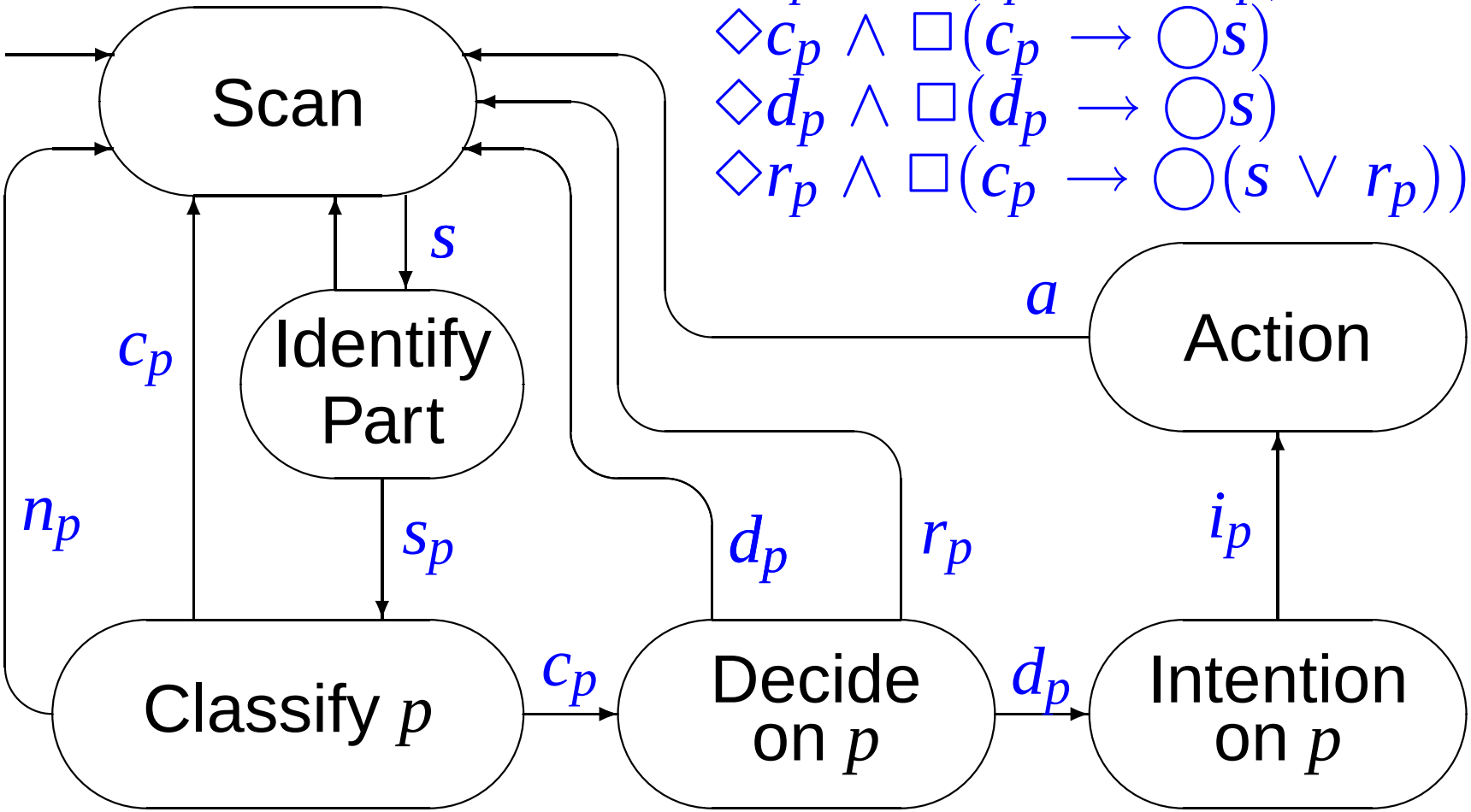
$\Box\neg\, i_p$ is decomposed as $\Box\neg\, s_p$

$\Diamond s_p \wedge \Box(s_p \rightarrow \bigcirc n_p)$

$\Diamond c_p \wedge \Box(c_p \rightarrow \bigcirc s)$

$\Diamond d_p \wedge \Box(d_p \rightarrow \bigcirc s)$

$\Diamond r_p \wedge \Box(c_p \rightarrow \bigcirc(s \vee r_p))$



Contrary Decision Process

$\Diamond r_p \wedge \Box(c_p \rightarrow \bigcirc(s \vee r_p))$

(phenotype error)

Possible genotype error is

- memory of previous decisions on similar pairs resulting in unnecessary actions

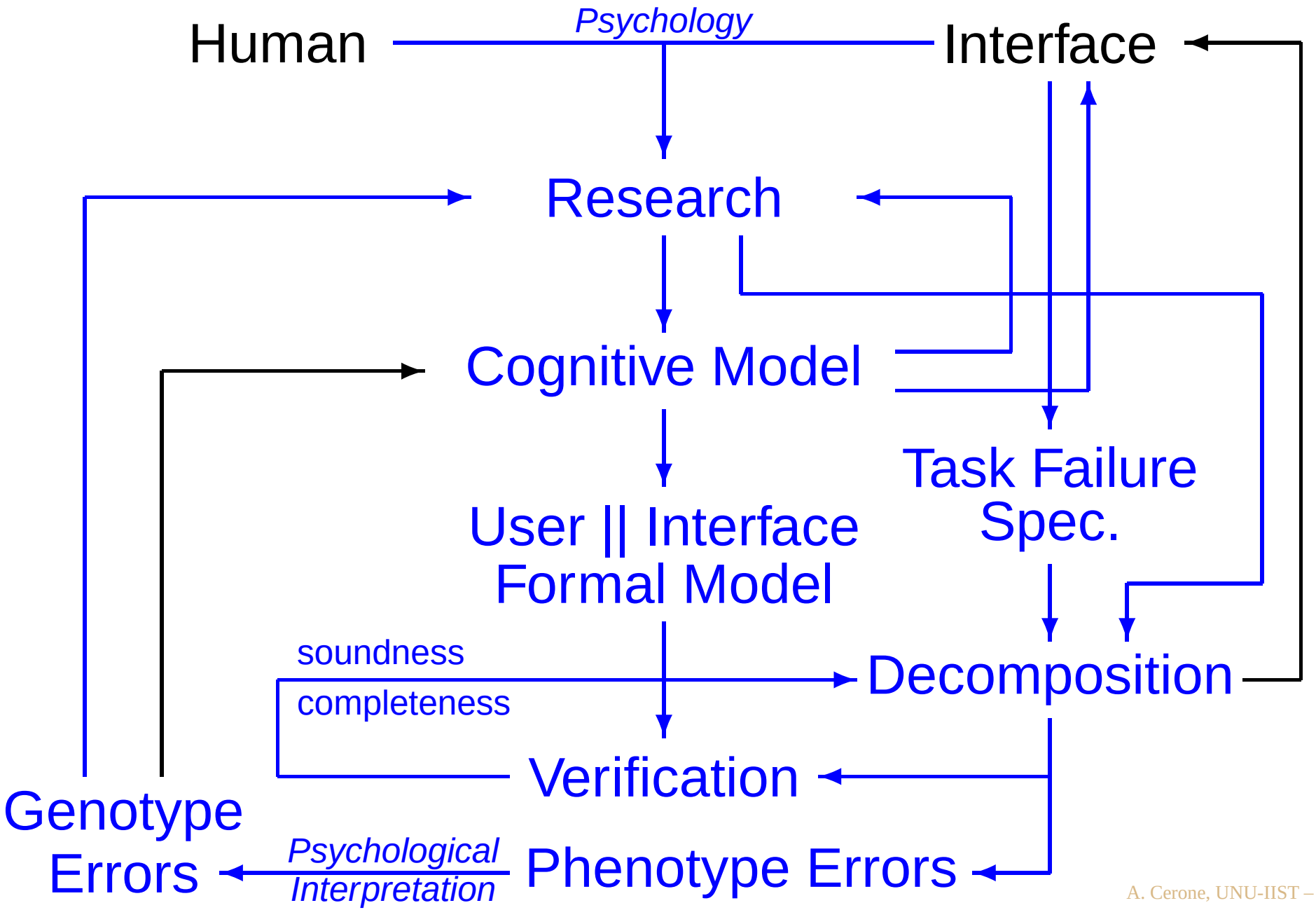
due to

- fear
- high workload

Error Cause

What caused such an error?

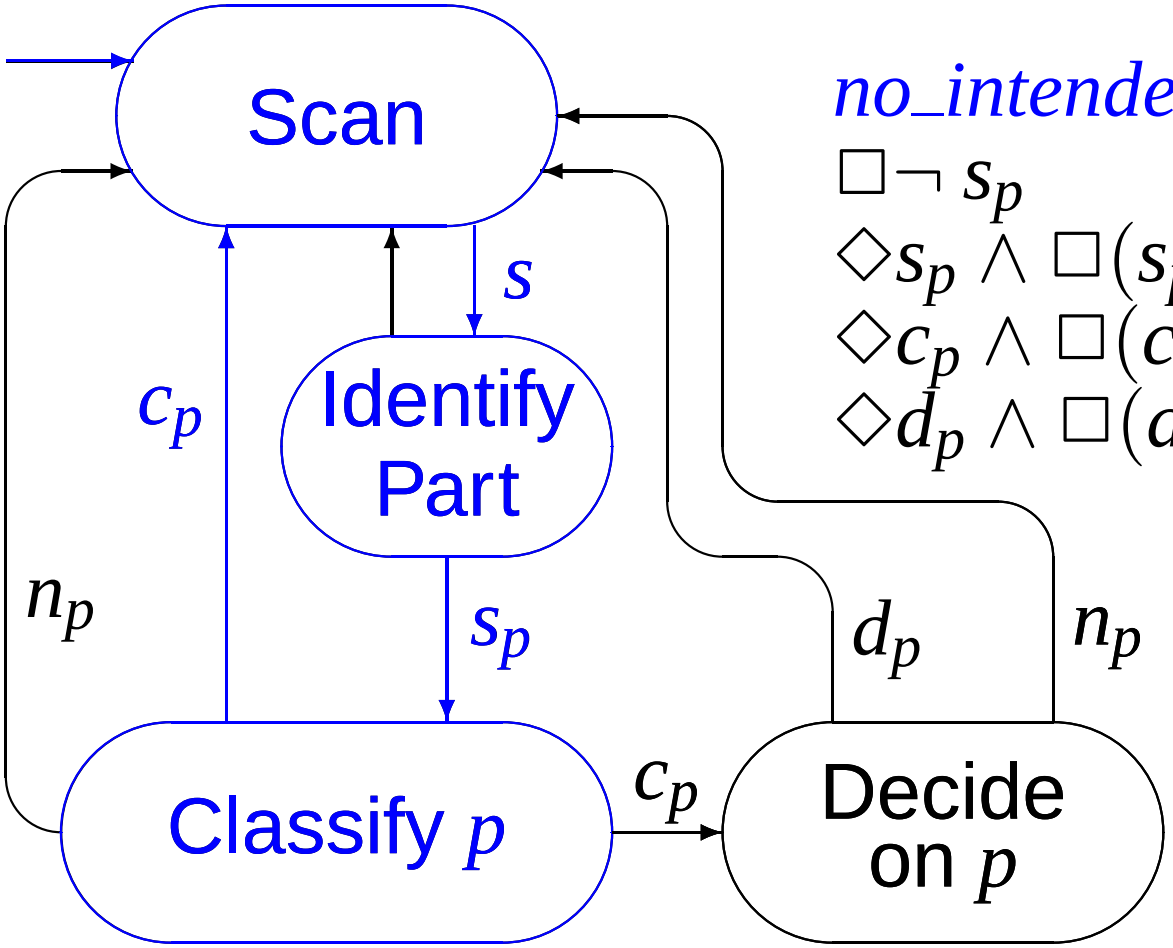
- use of the same action name n to denote the results of two cognitive processes
- aim at an elegant and easy to understand (to psychologists) formal model
  $\Longrightarrow$ focus on syntactical look of formulae
  rather than on their interpretation on the model


Human
Psychology
Interface
Research
Cognitive Model
Task Failure
Spec.
User || Interface
Formal Model
soundness
completeness
Decomposition
Verification
Genotype
Errors
Psychological
Interpretation
Phenotype Errors

Exercise: Counterexample?

Find and analyse the counterexample

$$s \longrightarrow s_p \longrightarrow c_p \longrightarrow s \longrightarrow s_p \longrightarrow c_p \longrightarrow n_p \longrightarrow s \longrightarrow \ldots$$



$no\_intended\_response_p$:

$\Box \neg s_p$

$\Diamond s_p \wedge \Box(s_p \vee c_p \rightarrow \bigcirc n_p)$

$\Diamond c_p \wedge \Box(c_p \rightarrow \bigcirc s)$

$\Diamond d_p \wedge \Box(d_p \rightarrow \bigcirc s)$

References

[Dix et al. 98]

Alan Dix, Janet Finaly, Gregory Abowd, Russel Beale.
Human-Computer Interaction.
Prentice Hall, 2nd Edition, 1998.

HCI Textbook

One of the most complete general textbooks in HCI, also introduces the use of seveal formal notations, such as Petri nets, CSP, temporal logic, Z. There is now a 3rd edition.
Complementary materials available online at

http://www.hiraeth.com/books/hci/

[Preece et al. 94]

Jenny Preece, Yvonne Rogers, Helen Sharp, David Benyon, Simon Holland and Tom Carey. *Human-Computer Interaction*. Addison Wesley, 1994.

HCI Textbook

The first HCI textbook to contain all pedagogical features (examples, exercises, etc.). Now a bit old. Book review available online at

http://www.acm.org/∼perlman/preece.html

[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.

[Dix 91]

Alan Dix.
Formal Methods for Interactive Systems.
Academic Press, 1991.

FMIS Textbook
Out of print, but available online at
http://www.hiraeth.com/books/formal/

[FMIS 06]

A. Cerone and P. Curzon.
Proceedings of FMIS 2006.
ENTCS 183, Elsevier, 2007

Extended version of a selection of the papers has been published in
Software and System Modeling Vol. 4, No. 2, Springer, 2008

[FMIS 07]

A. Cerone and P. Curzon.
Proceedings of FMIS 2007.
ENTCS , Elsevier, 2007

Extended version of a selection of the papers has been published in
Formal Aspects of Computing Vol. 21, No. 6, Springer, 2009

[Cerone and Elgegbyan 07]

A. Cerone and N. Elgegbyan.
Model-checking Driven Design of Interactive Systems.
ENTCS 183, Elsevier, 2007, pages 3–20.

Formal Methods Paper
Use of model-checking to improve the interface design with respect to security properties.

[Lindsay et. al. 02]

P. Lindsay and S. Connelly.
Modelling Erroneous Operator Behaviour for an Air-traffic Control Task.
AUIC 2002, Conferences in Research and Practice in Information Technology, Vol. 7, Australian Computer Society, pages 43–54.

Formal Methods Paper
Incorrect decomposition for the ATC Example.

[Cerone et al. 05 and 08]

A. Cerone, P. Lindsay and S. Connelly.
Formal Analysis of Human-computer Interaction using Model-checking.
SEFM 2005, IEEE Comp. Soc., 2005, pages 352–361.

A. Cerone, S. Connelly and P. Lindsay.
Formal Analysis of Human Operator Behavioural Patterns in Interactive Surveillance Systems.
Software and Systems Modeling, Vol.7, No. 3, Springer, 2008, pages 273–286.

Formal Methods Paper
On the correct decomposition for the ATC Example, but the second is the most complete.

End