Phd course on

Formal modelling and analysis of interactive systems

Part 3
Formal Analysis and Cognitive Models

Temporal Logic, Formal Analysis of 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

Modal and Temporal Logic

Propositional Logic

$p =$ the train is late
$q =$ there are taxis at the station
$r =$ John is late for his meeting

if the train is late
  and there are
    not taxis at the station
then John is late for his meeting

the train is late
  $\wedge \neg$ there are taxis at the station
  $\rightarrow$ John is late for his meeting

$p \wedge \neg q \rightarrow r$

Propositional Logic II

$p =$ it is raining
$q =$ Jane has her umbrella with her
$r =$ Jane gets wet

if it is raining
  and Jane has
    not her umbrella with her
then Jane gets wet

it is raining
  $\wedge \neg$ Jane has her umbrella with her
  $\rightarrow$ Jane gets wet

$p \wedge \neg q \rightarrow r$

Predicate Logic

Every child is younger than its mother

$C(x) = x$ is a child
$Y(x,y) = x$ is younger than y
$M(x,y) = x$ is mother of y

Predicates

$\forall$ — universal quantifiers
$\exists$ — existential quantifiers

$$\forall x \forall y (C(x) \wedge M(y,x) \rightarrow Y(x,y))$$

Prop vs Pred Logic

- Propositional Logic
  - decidability
  - limited expressiveness
- Predicate Logic
  - undecidability
  - good expressiveness

Modal Logic

Every child is younger than its mother

true in all families

Every niece is younger than her uncle

not necessarily true in all families
possibly not true in all families

$\Box f$ expresses that f is necessary
$\Diamond f$ expresses that f is possible

Property: $\neg \Diamond f = \Box \neg f$

Expressiveness: between Prop and Pred Logic
Decidable!

Modal Logic: Semantics

Kripke Model

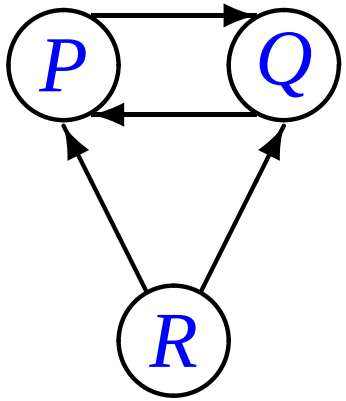
$$\langle \mathcal{W}, \mathcal{R}, \mathcal{L} \rangle$$

such that

- $\mathcal{W}$ is a set of worlds
- $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$ is the accessibility relation
- $\mathcal{L} : \mathcal{W} \longrightarrow 2^{AP}$ is the labelling function

$\mathcal{L}(R) = \{f, g\}$
$\mathcal{L}(P) = \{f\}$
$\mathcal{L}(Q) = \{f, h\}$

$\Diamond g$ and $\Box f$ are valid
$\Box g$ and $\Box h$ are not valid



Modal $\longrightarrow$ Temporal Logic

Transition System

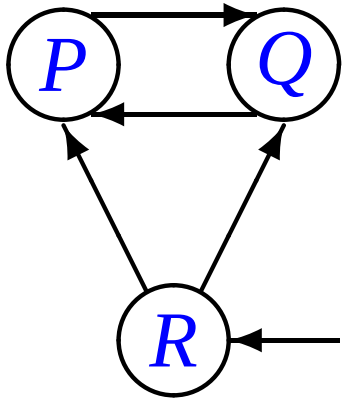
$$\langle S, \rightarrow, s_0 \rangle$$

such that

- S is a set of states
- $\rightarrow \subseteq S \times S$ is the transition relation
- $s_0 \in S$ is the initial state

Labelling function: $\mathcal{L} : S \longrightarrow 2^{AP}$

$\mathcal{L}(R) = \{f, g\}$
$\mathcal{L}(P) = \{f\}$
$\mathcal{L}(Q) = \{f, h\}$
$\Diamond g$ and $\Box f$ are valid
$\Box g$ and $\Box h$ are not valid



Temporal Logic of Actions

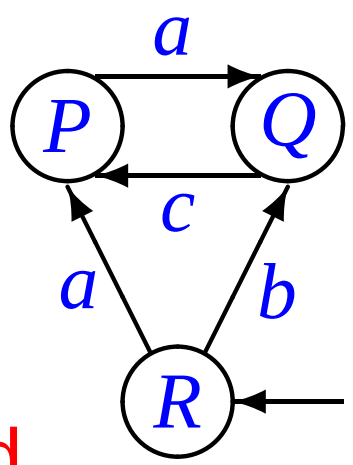
Labelled Transition System

$$\langle S, L, \rightarrow, s_0 \rangle$$

such that

- S is a set of states
- $\mathcal{L}$ is a set of labels
- $\rightarrow \subseteq S \times \mathcal{L} \times S$ is the transition relation
- $s_0 \in S$ is the initial state

$AP = L$



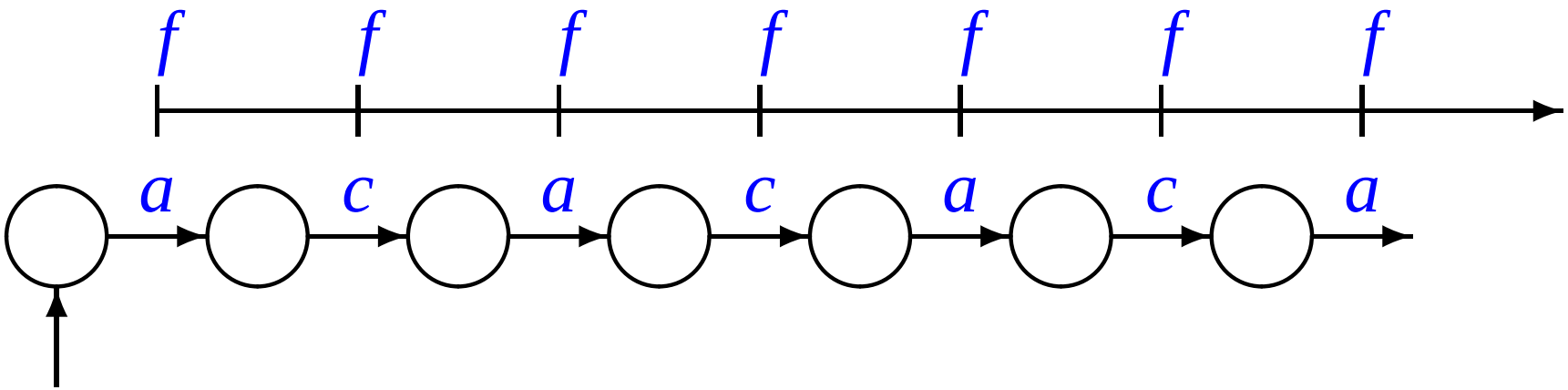
$\Diamond a$ and $\Diamond c$ are valid
$\Diamond b$, $\Box a$, $\Box b$ and $\Box c$ are not valid

Liner Time Logic (LTL)

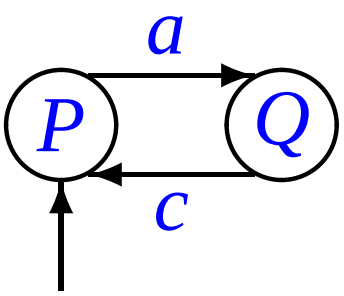
Linear Time:

LTL: "always" Operator

Linear Time:



$\Box f$ expresses that f is *always* true in the future



$P = a \rightarrow Q$

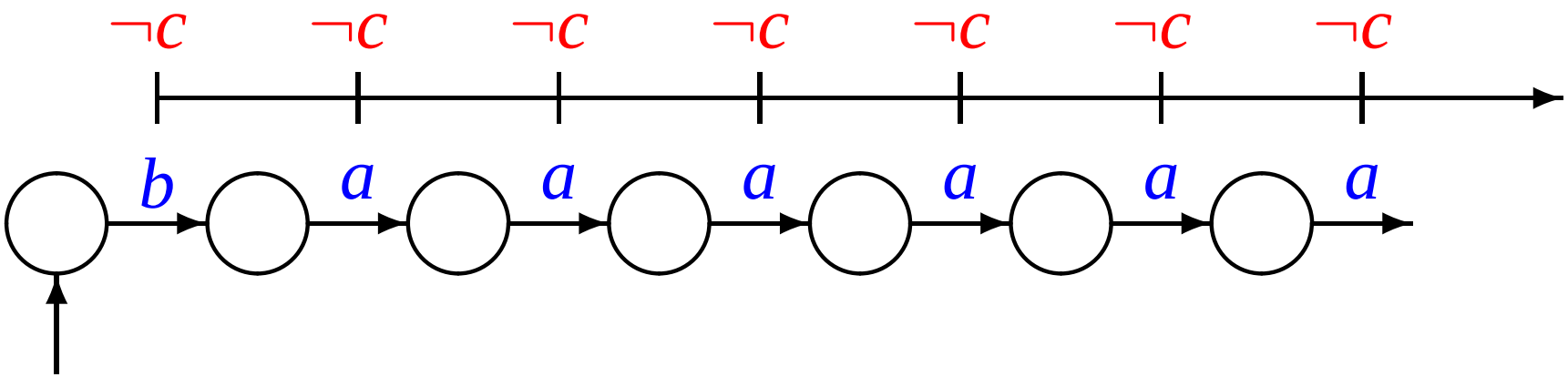
$Q = c \rightarrow P$

$\Box(a \vee c)$ is true

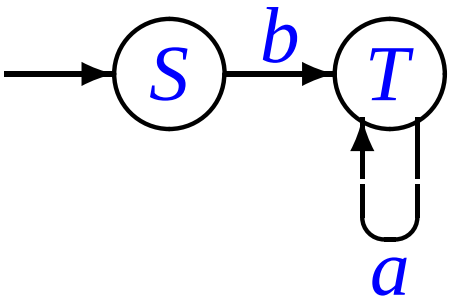
$\Box a$ is false

LTL: "eventually" Operator

Linear Time:



$\Diamond f$ expresses that f is *eventually* true in the future



$S = b \rightarrow T$

$T = a \rightarrow T$

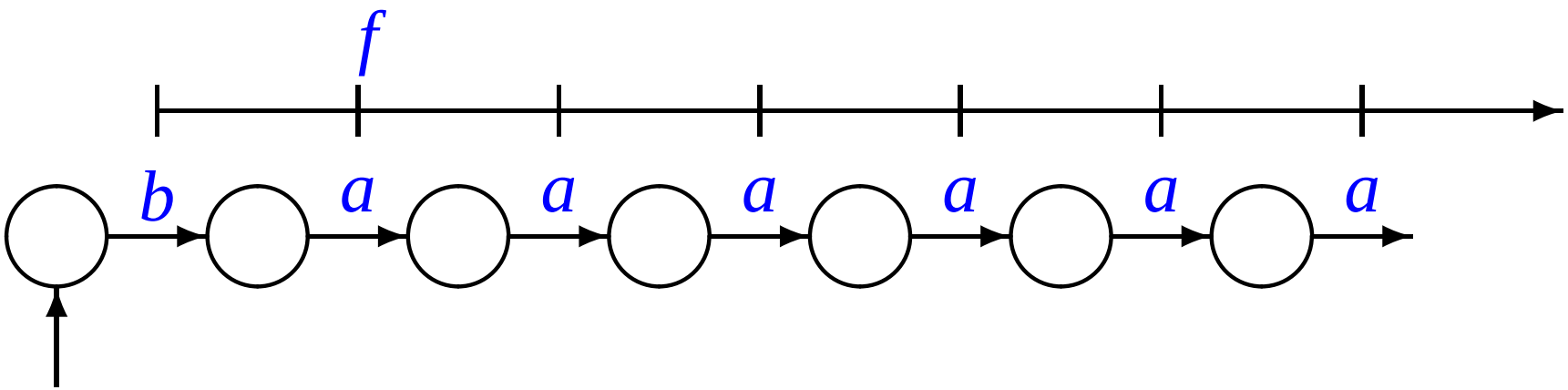
$\Diamond b$ is true

$\Diamond c$ is false
but $\Box \neg c$ is true

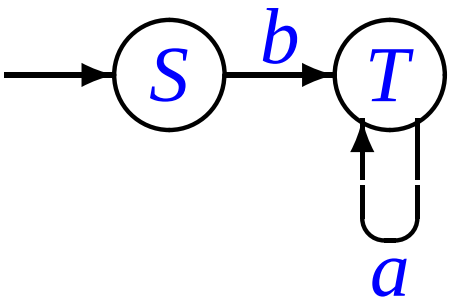
Property: $\neg \Diamond f = \Box \neg f$

LTL: "next" Operator

Linear Time:



$\bigcirc f$ expresses that f is is true at the *next state*



$S = b \rightarrow T$

$T = a \rightarrow T$

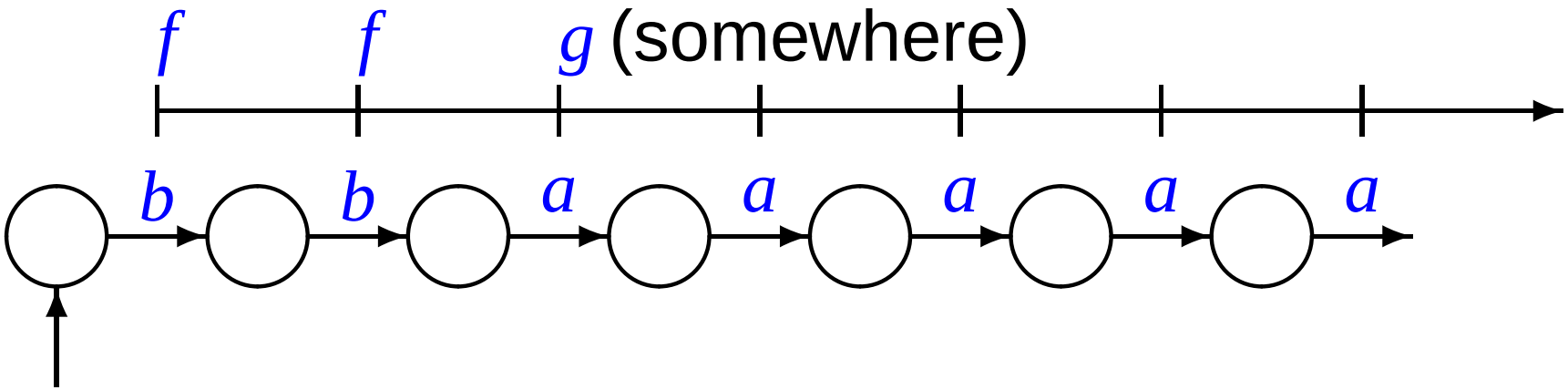
$\bigcirc a$ is true

$\bigcirc \Box a$ is true

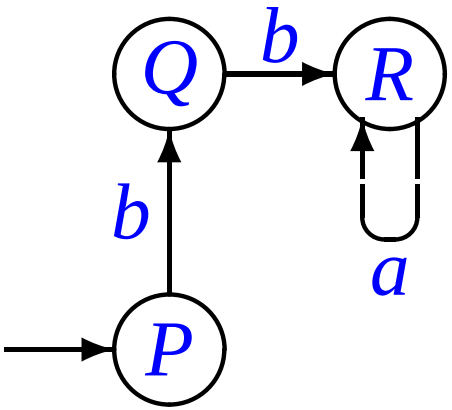
$\bigcirc b$ is false

LTL: "strong until" Operator

Linear Time:



$f \mathcal{U} g$ expresses that f is *always* true *until* g is true



$S = b \rightarrow Q$

$Q = b \rightarrow R$

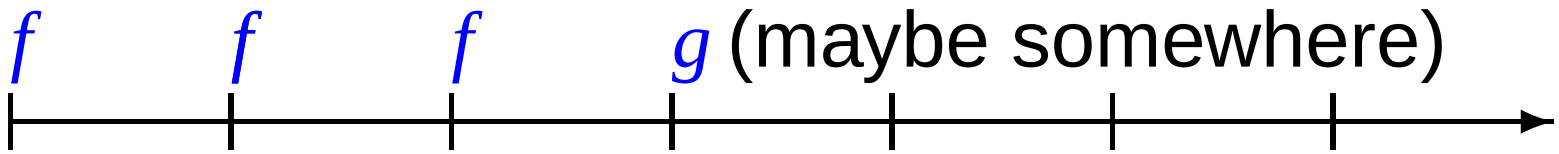
$R = a \rightarrow R$

$b \mathcal{U} a$ is true

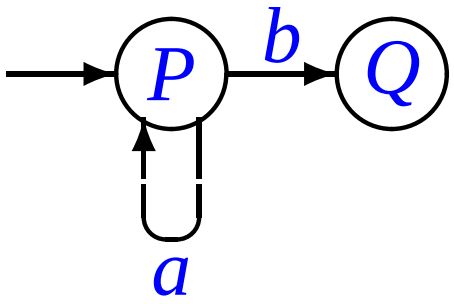
$b \mathcal{U} (\Box a)$ is true

LTL: "weak until" Operator

Linear Time:



$f \mathcal{W} g$ expresses that f is *always* true
either *forever* or *until* g is true



$P = (a \rightarrow p) [] (b \rightarrow Q)$ $a \mathcal{W} b$ is true

$Q = a \rightarrow Q$

Property: $(a \mathcal{U} b) \vee \Box a = a \mathcal{W} b$

Linear Time Logic (LTL)

- $\Box f$ expresses that f is *always* true in the future
- $\Diamond f$ expresses that f is *eventually* true in the future
- $\bigcirc f$ expresses that f is true at the *next state*
- $f \,\mathcal{U}\, g$ expresses that f is *always* true *until* g is true (strong until)
- $f \,\mathcal{W}\, g$ expresses that f is *always* true either *forever* or *until* g is true (weak until)

Properties:

$\neg \Diamond f = \Box \neg f$

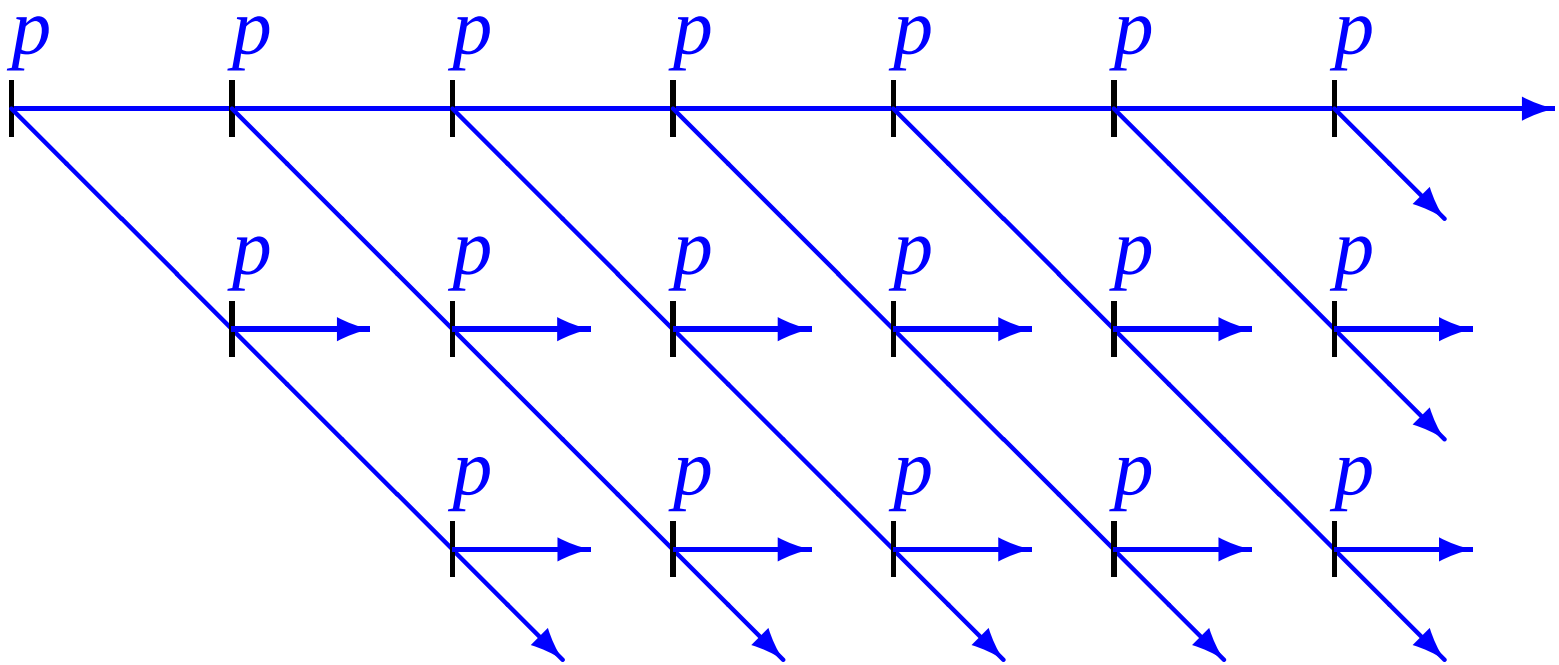
$f \,\mathcal{W}\, g = (\Box f) \vee (f \,\mathcal{U}\, g)$

Computation Tree Logic (CTL)

- branching-time logic
  $\Longrightarrow$ different paths in the future
  $\Longrightarrow$ future is not determined
- each temporal operator is associated with a quantifier on path: $\forall$ or $\exists$

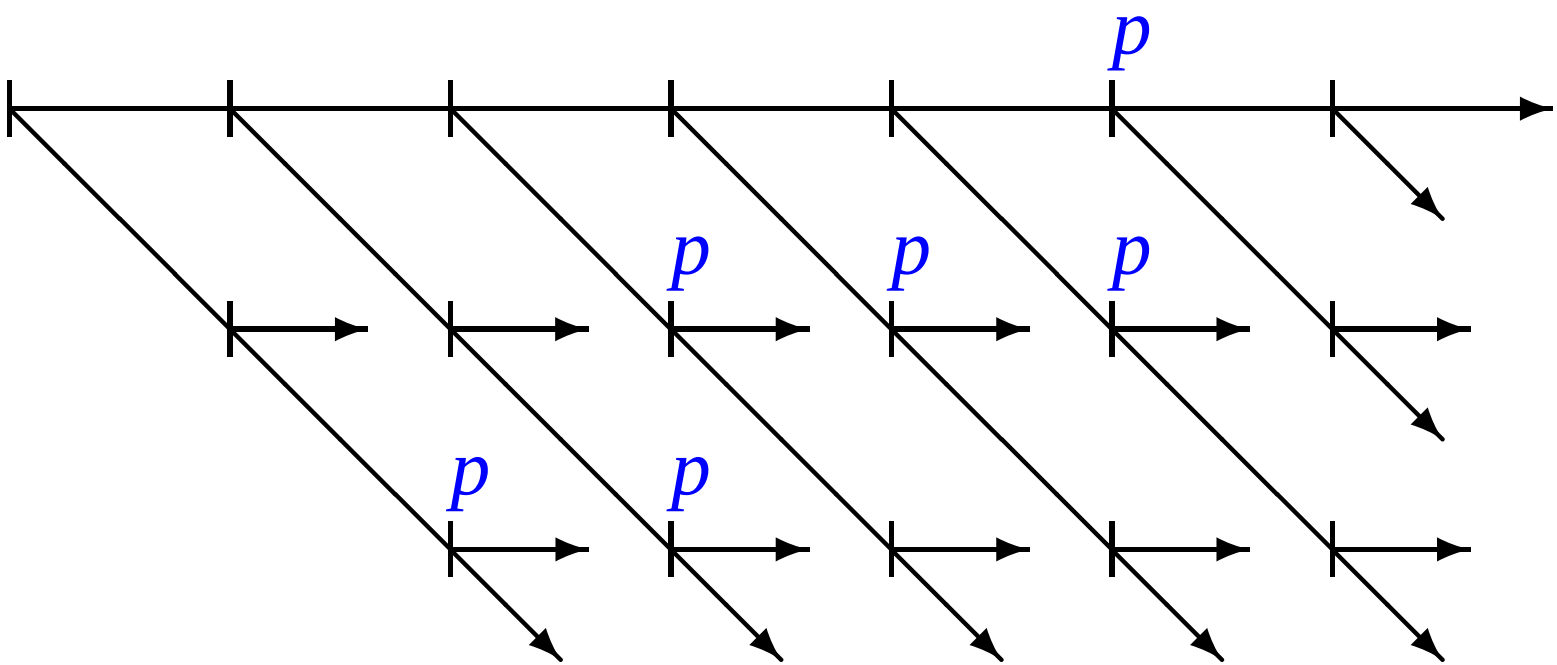
CTL: $\forall\Box$

$\forall\Box p$



CTL: $\forall \Diamond$

$\forall \Diamond p$



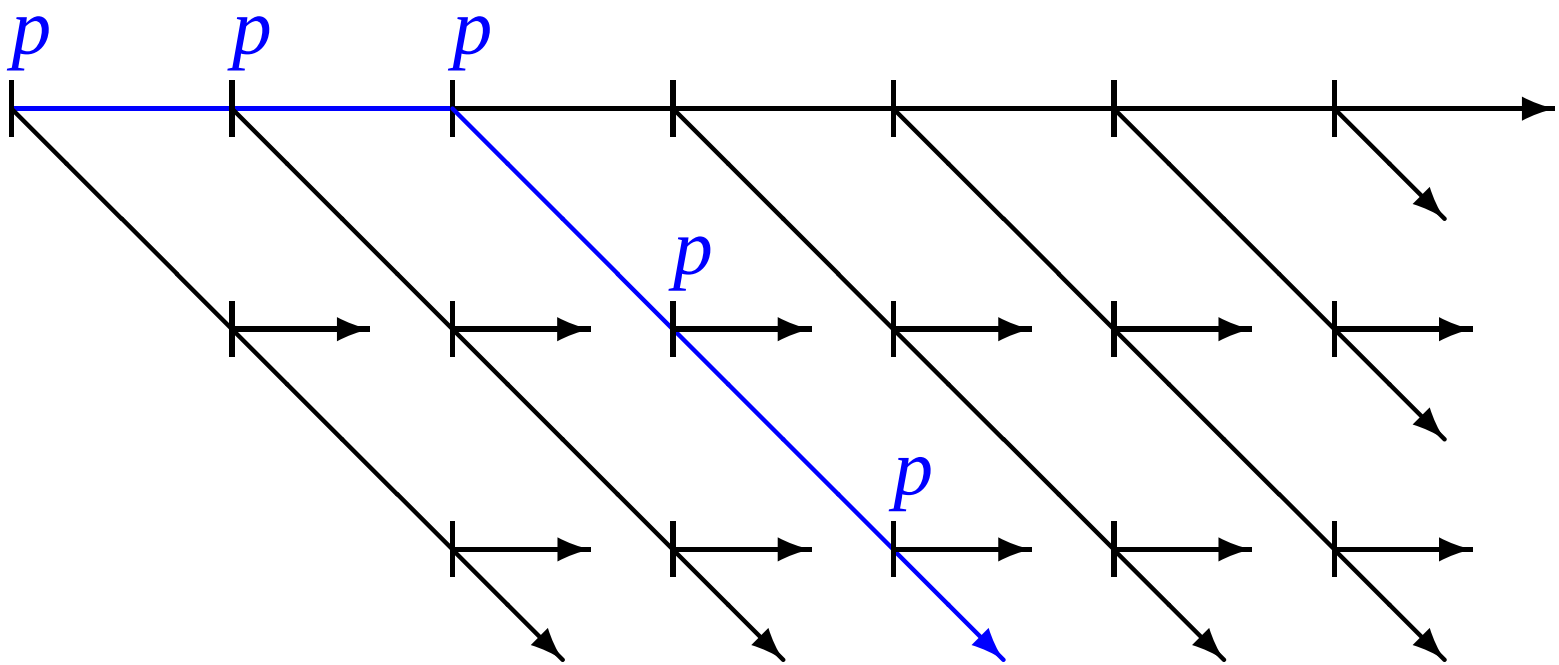
$CTL: \forall$

For all path

- f is *always* true: $\forall \Box f$
- f is *eventually* true: $\forall \Diamond f$
- f is true at the *next state*: $\forall \bigcirc f$
- f is *always* true *until* g is true: $\forall (f \mathcal{U} g)$
- f is *always* true either *forever* or *until* g is true: $\forall (f \mathcal{W} g)$

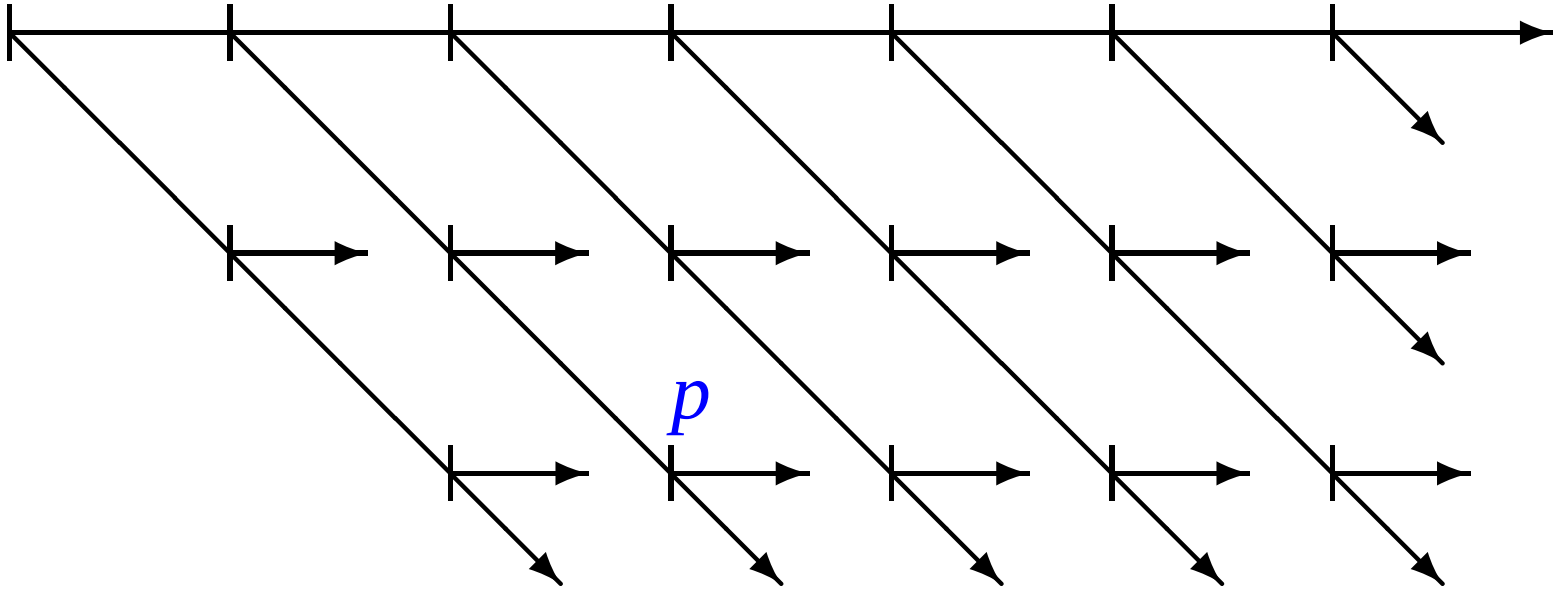
CTL: $\exists \Box$

$\exists \Box p$



CTL: $\exists \Diamond$

$\exists \Diamond p$



$CTL: \exists$

There exists a path

- f is *always* true: $\exists \Box f$
- f is *eventually* true: $\exists \Diamond f$
- f is true at the *next state*: $\exists \bigcirc f$
- f is *always* true *until* g is true: $\exists (f \, \mathcal{U} g)$
- f is *always* true either *forever* or *until* g is true: $\exists (f \, \mathcal{W} g)$

Properties:

$\neg \exists \Diamond f = \forall \Box \neg f$

$\exists (f \, \mathcal{W} g) = (\exists \Box f) \vee (\exists f \, \mathcal{U} g)$

*CTL**

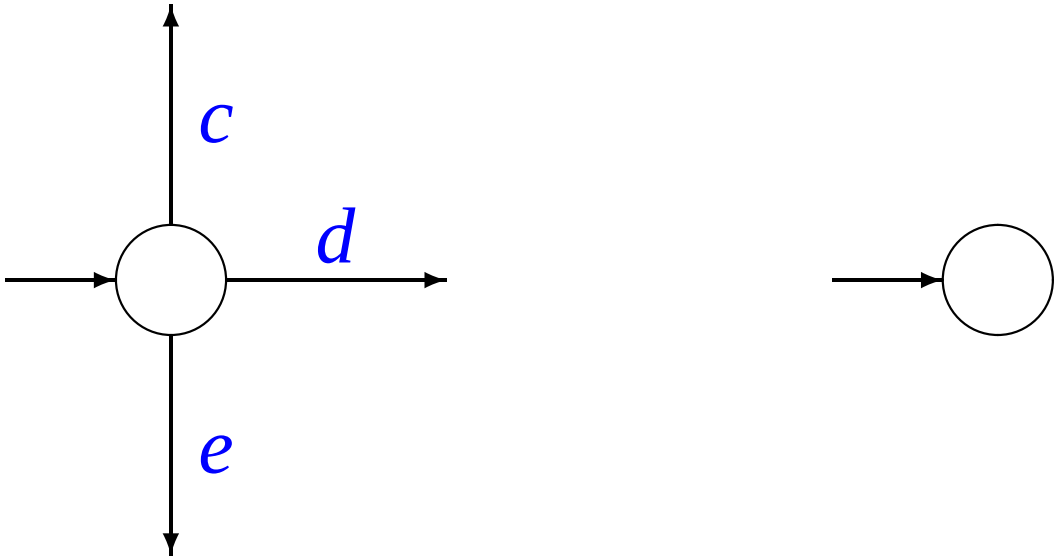
- Path Formula $=$ LTL Formula
- Path Formula $\Longrightarrow$ State Formula
- f State Formula $\Longrightarrow \forall f$ State Formula
- f State Formula $\Longrightarrow \exists f$ State Formula

GCTL in CWB-NC*

- Atomic Formulae:
  - `{ p1,...,pn }`
  - `{ -p1,...,pn }` (deadlock-free: `{ -}`)
- $\neg, \vee, \wedge$: `~, \/, /\`
- $\Box f$: `Gf`
- $\Diamond f$: `Ff`
- $\bigcirc f$: `Xf`
- $\forall f$: `Af`
- $\exists f$: `Ef`

LTL formula `f` is expressed in GCTL* as `Af`

*Negation in GCTL**



true $\sim$ { a,b } true

true { -a,b } false

true { -} false

ATM Formal Specification

ATM Properties in LTL

Functional Correctness:

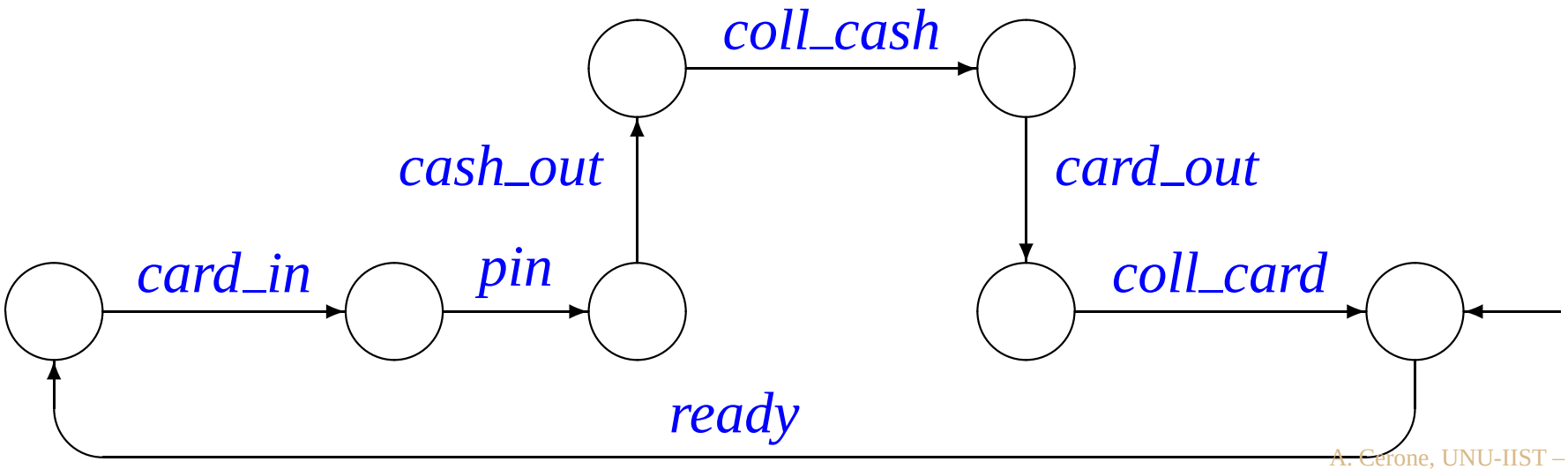
The ATM machine will eventually deliver cash

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

Safety:

The ATM machine will eventually return the card

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



*ATM Properties in CTL**

Functional Correctness:

The ATM machine will eventually deliver cash

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

Safety:

The ATM machine will eventually return the card

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

Express the properties above in CTL*

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

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

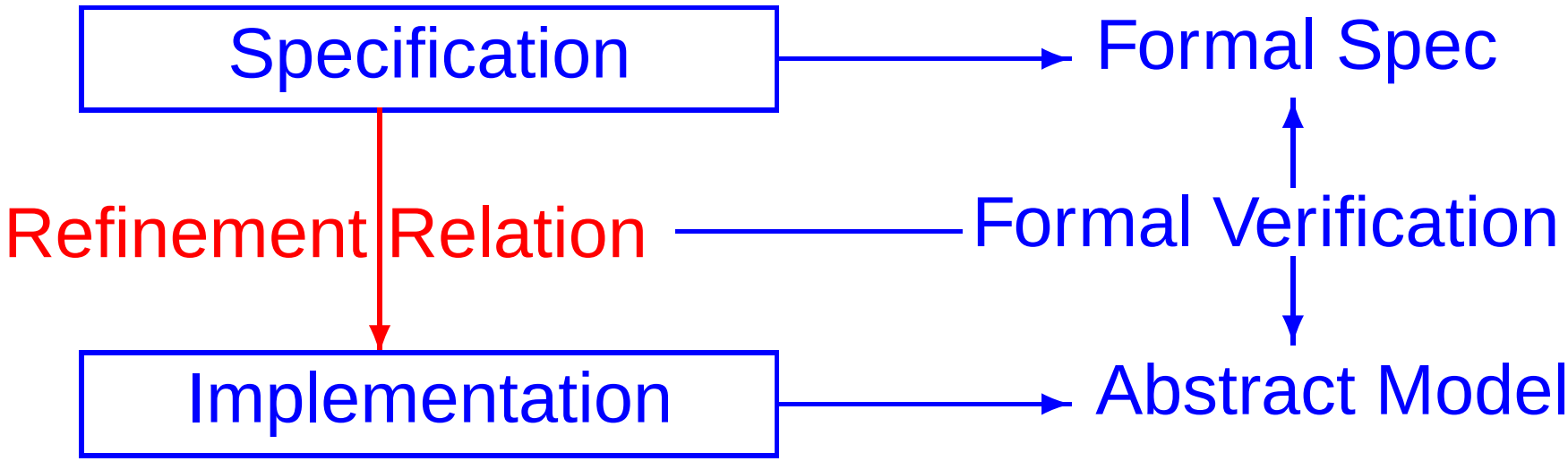
  $\forall \Box (card\_out \rightarrow ((\neg ready) \,\mathcal{U}\, coll\_card))$

Formal Analysis

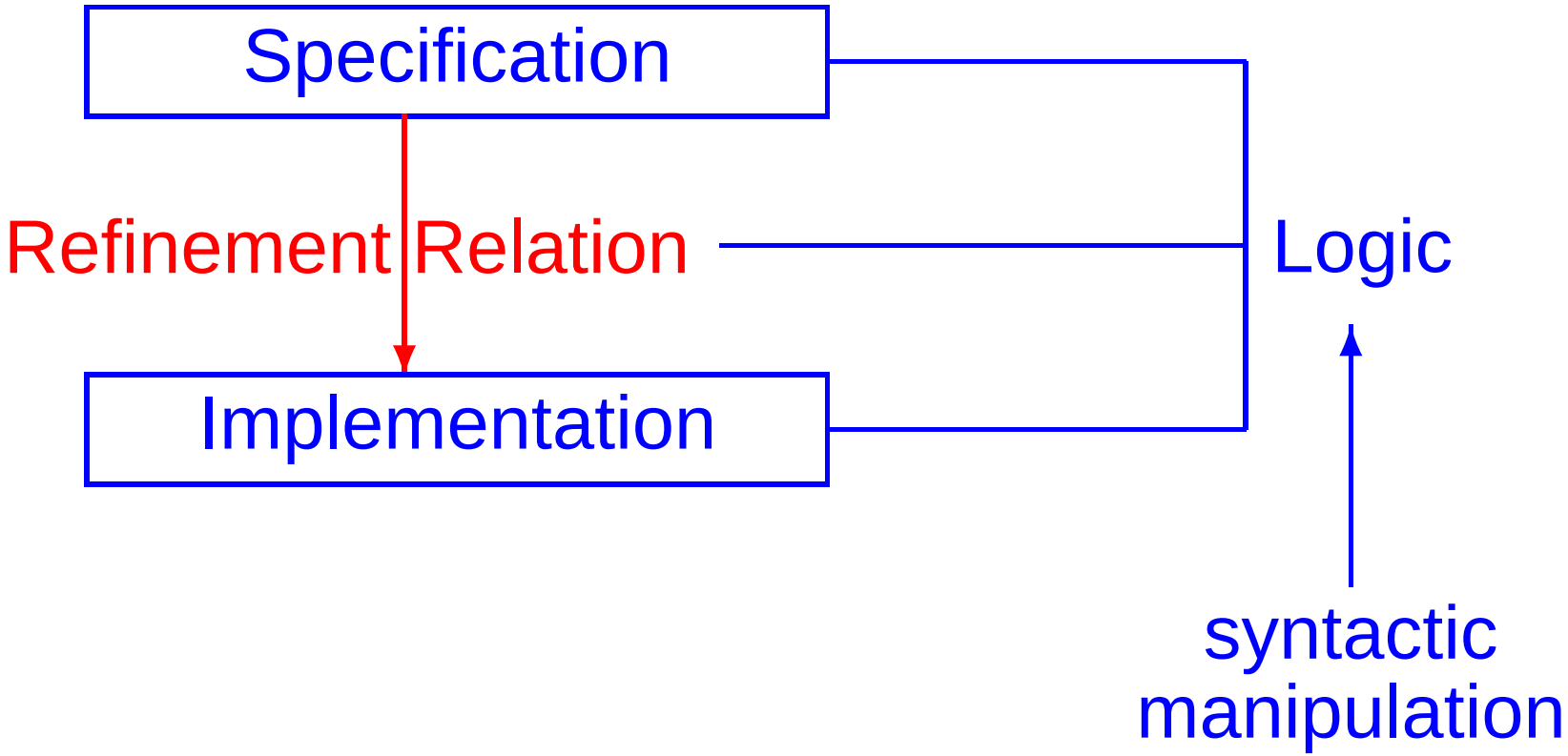
Simulation and Verification

- Started from an Informal Specification
- $\Longrightarrow$ Formal Model
  - abstract form of Implementation
  - debugged using Simulation (Analysis)
- $\Longrightarrow$ Formal Specification
  - unambiguous form of Specification
- Analysis
  - Formal Verification of the Model against the Specification

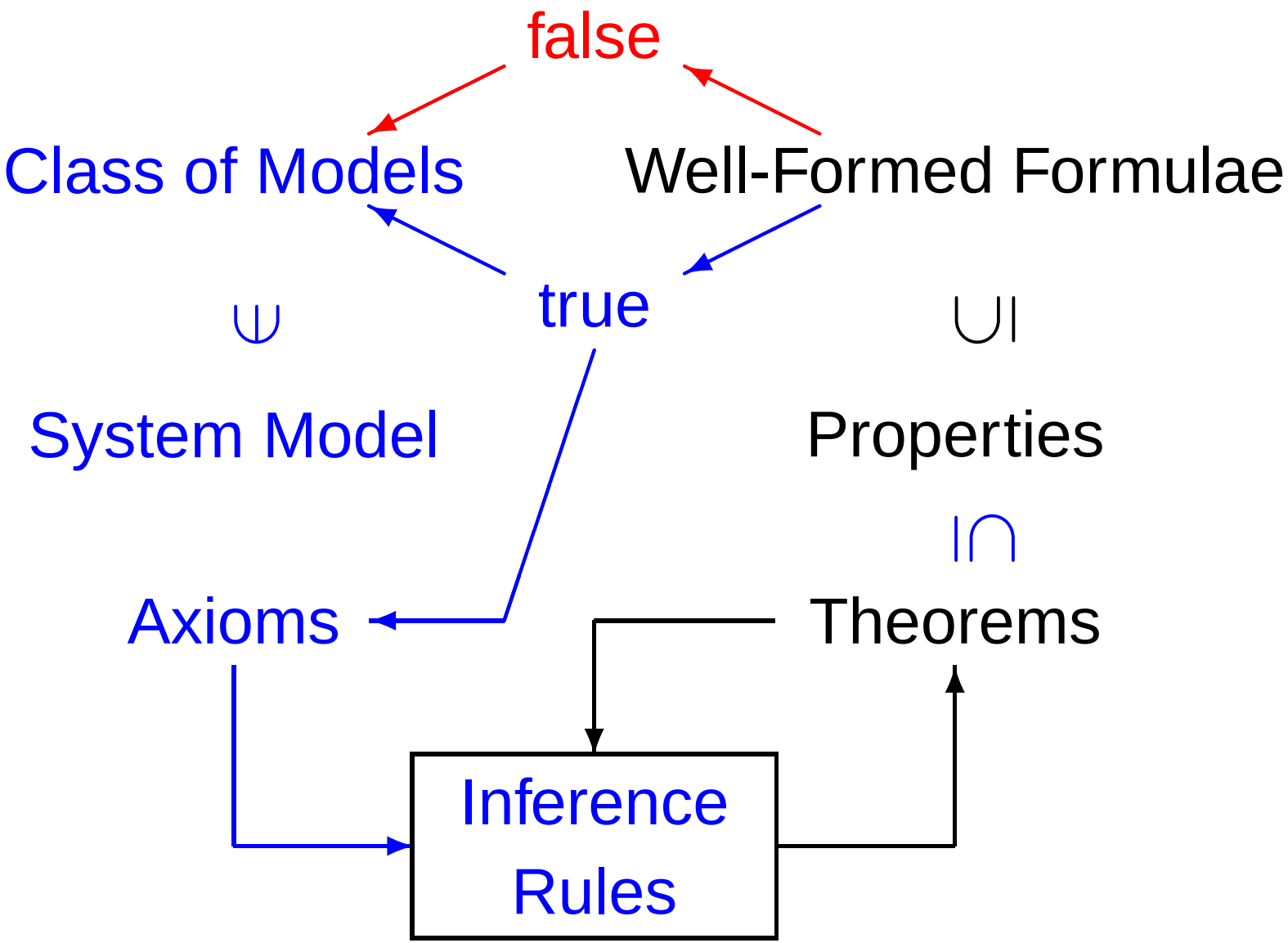
Formal Verification



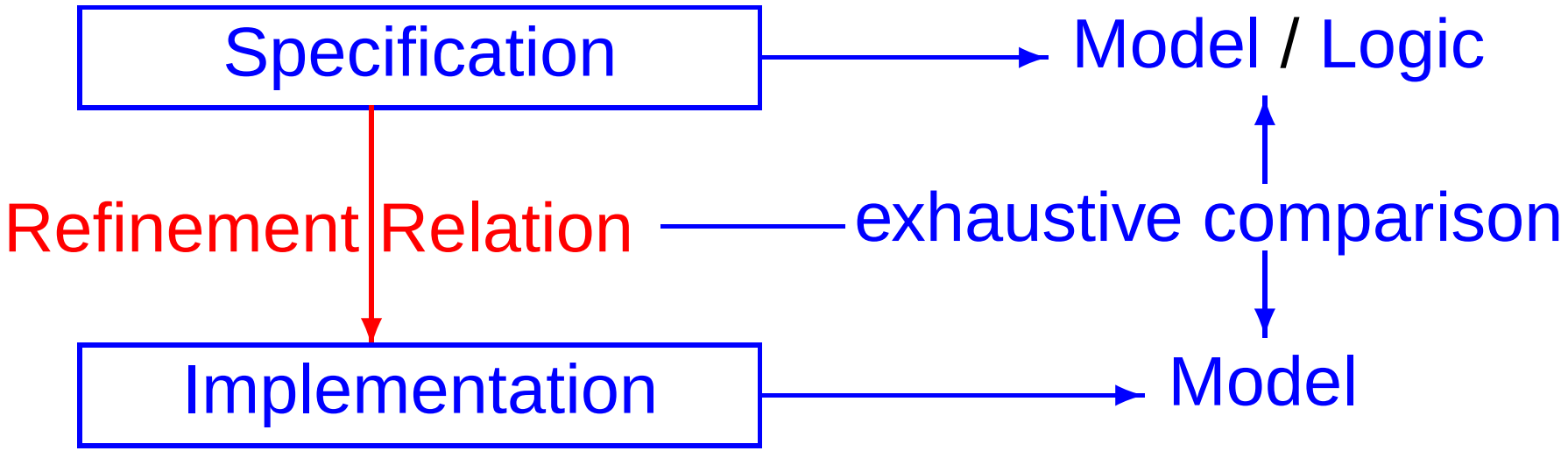
Theorem-Proving



How Theorem-proving Works



Model-Checking



Theorem-proving: Pros & Cons

- Advantages
  - Maximum Modelling Expressivity
    - Infinite State Systems
    - Complex data structure
- Disadvantages
  - Semidecidability
  - Verification procedure not fully automated
  - Tools difficult to use
  - No scalability
  - Does not allow debugging

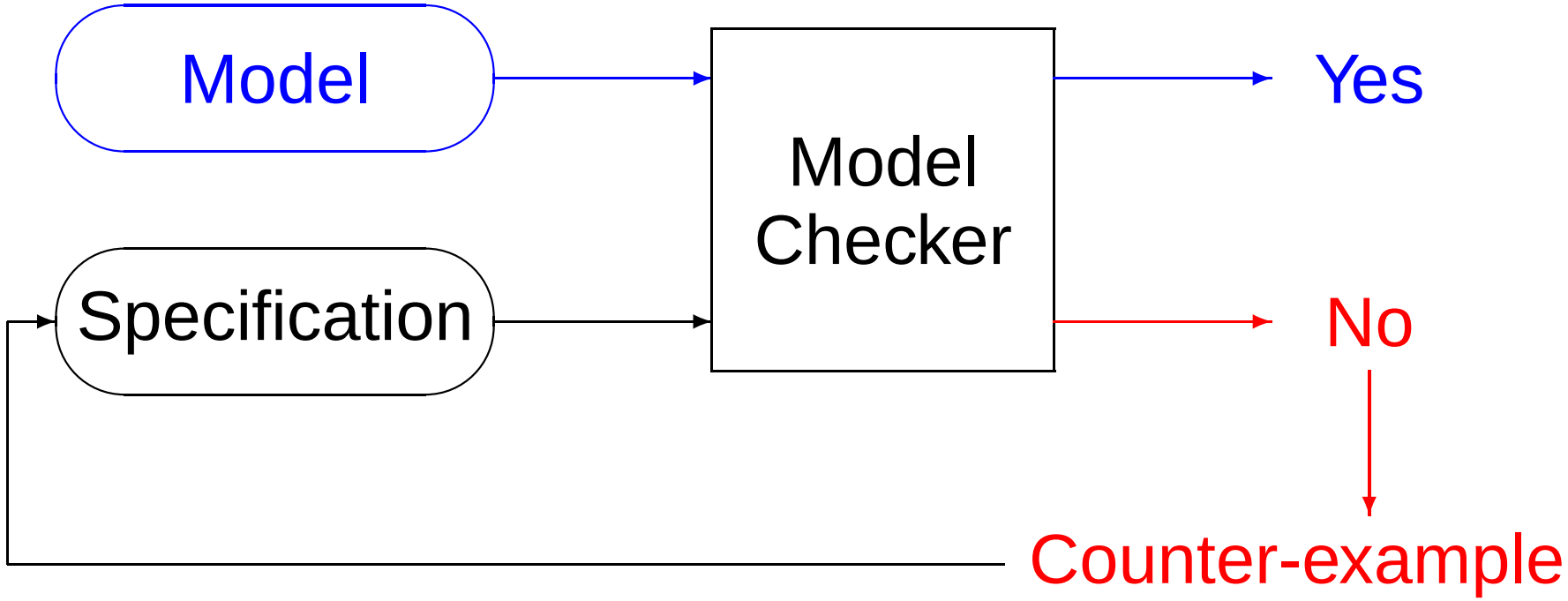
Model-Checking: Pros & Cons

- Advantages
  - Decidability
  - May be fully automated
  - Better usability tools
  - Good scalability
  - Allows debugging
- Disadvantages
  - Limited Expressivity
    - Finite State Systems
    - Limited data structure

History of Model-checking

- 1980s: Model-checking
  - *[Emerson and Clarke]*, *[Sifakis]*
  - Hardware Verification
  - State Explosion Problem
- 1990s:
  - Symbolic Model-checking *[MacMillan]*
  - Abstraction
  - State Explosion Contained
  - Infinite Model-checking
  - Software Verification

Model-checking



Model-checkers

- CWB-NC
  Concurrency Workbench of the New Century
  http://www.cs.sunysb.edu/ cwb/
- SAL
  Symbolic Analysis Laboratory
  http://sal.csl.sri.com/
- SMV
  http://www.cs.cmu.edu/ modelcheck/smv.html
- SPIN
  http://spinroot.com/
- FDR
  http://www.fsel.com/

Model-checking Demo

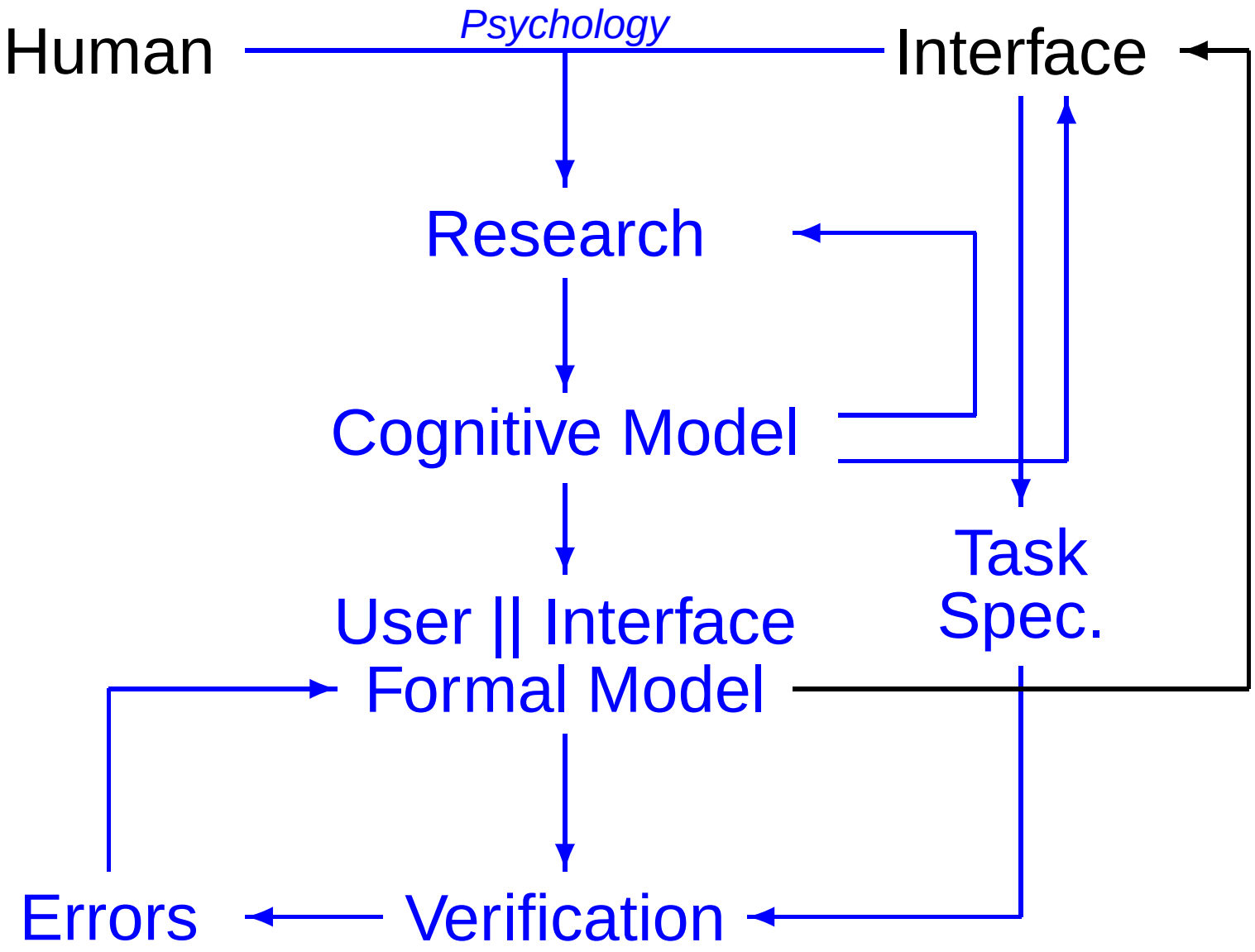
Analysis of Interactive Systems

Simulation and Verification

- Started from an Informal Specification
- $\Longrightarrow$ Formal Model
  - abstract form of Implementation
  - debugged using Simulation (Analysis)
- $\Longrightarrow$ Formal Specification
  - unambiguous form of Specification
- Analysis
  - Formal Verification of the Model against the Specification

Verifying Interactive Systems

- Started from an Informal Specification
- $\Longrightarrow$ Formal Model
  - Interface (Machine) $\Longleftarrow$ Implementation
  - Human (User) $=$ Cognitive Model
- $\Longrightarrow$ Formal Specification
  - unambiguous form of Task Specification
- Analysis
  - Formal Verification of the Interface in the presence of the Cognitive Model against the Specification


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

Model of Attention

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

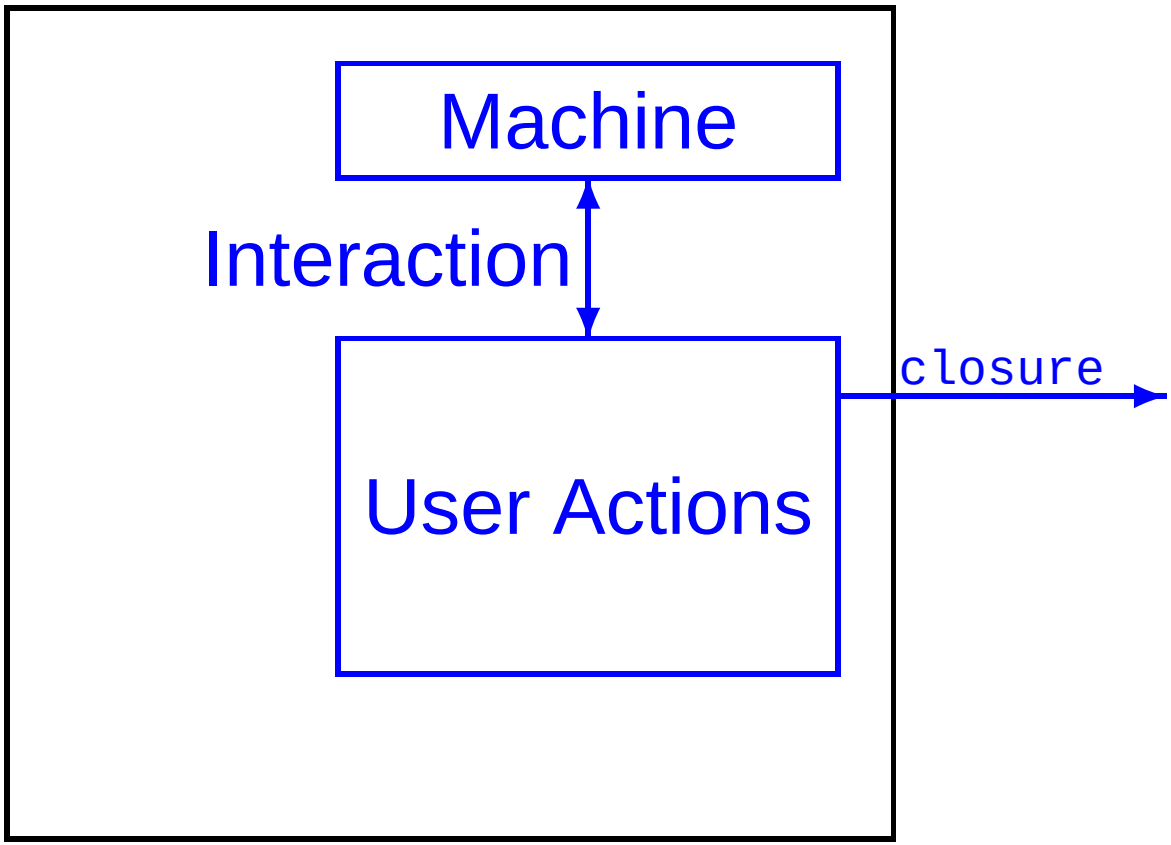
- required decision
  selections: kind of transaction, print balance
- danger
  card returned unexpectedly
- novelty
  keyboard on the screen,
  cash given at earlier stage
- temptation
  message: enter a draw if you withdraw ...
- anger
  message: no cash available

SAS activation in ATM (cont)

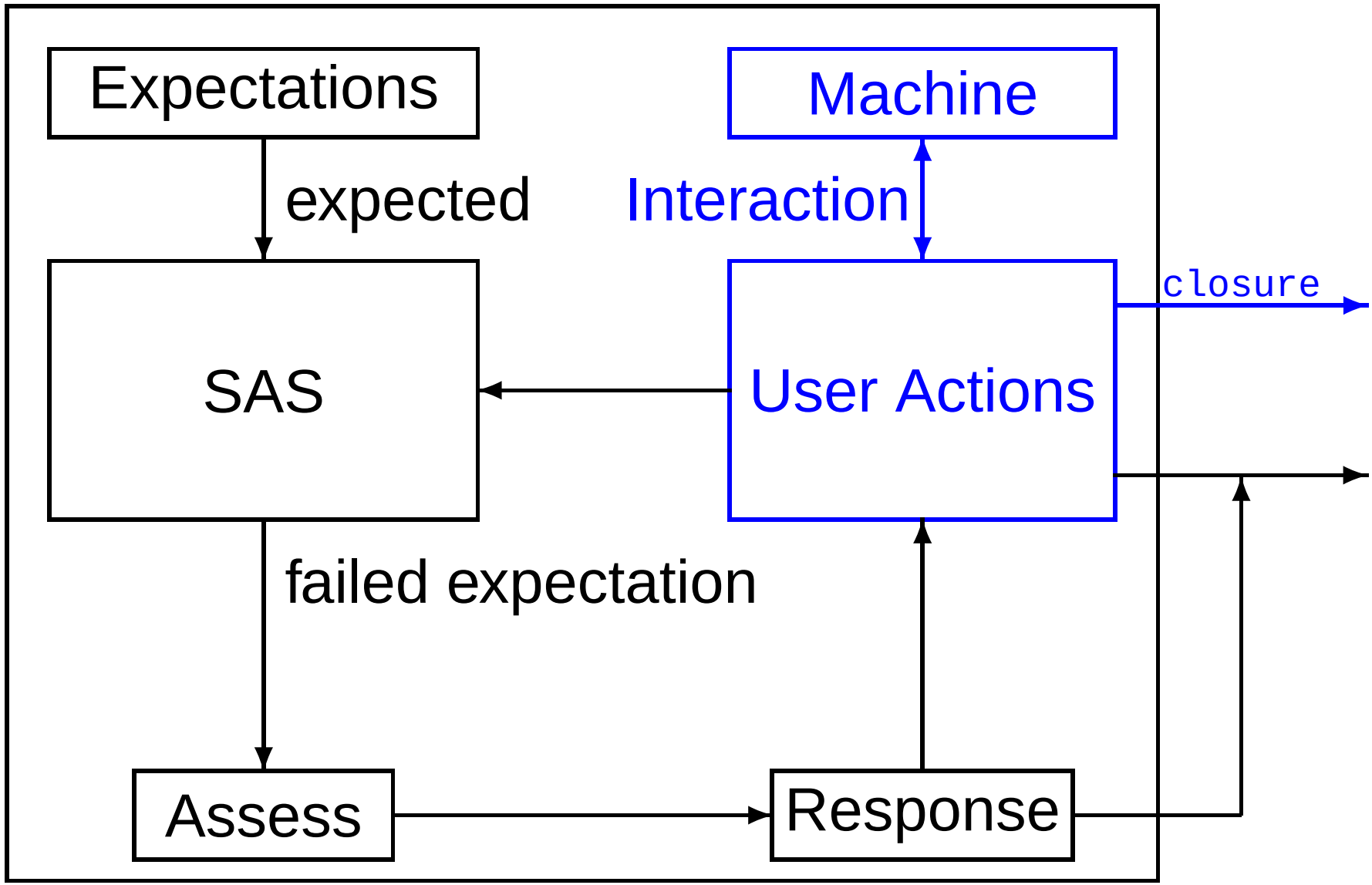
- required decision
  $\Longleftarrow$ choice operator
- danger
  $\Longleftarrow$ danger response = leave the interaction
- novelty
  $\Longleftarrow$ depends on the specific situation

ATM Example Revisited

Interaction and Closure



SAS in ATM


Expectations
Machine
expected
Interaction
closure
SAS
User Actions
failed expectation
Assess
Response

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

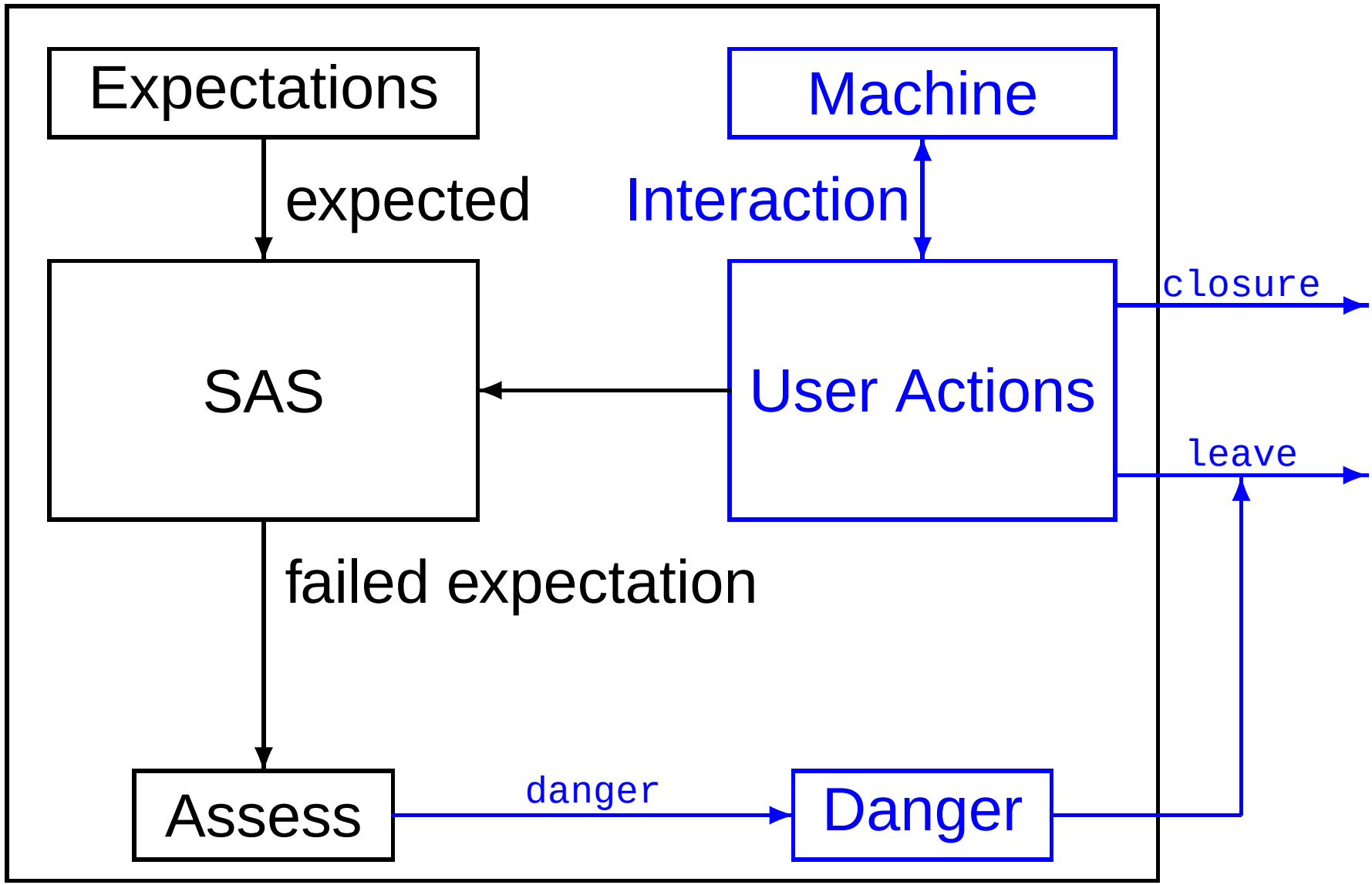
```
proc CollCardStart =
    start_int -> CollCardToDo
    card_out -> CollCardStart

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
```

SAS in ATM: Danger



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

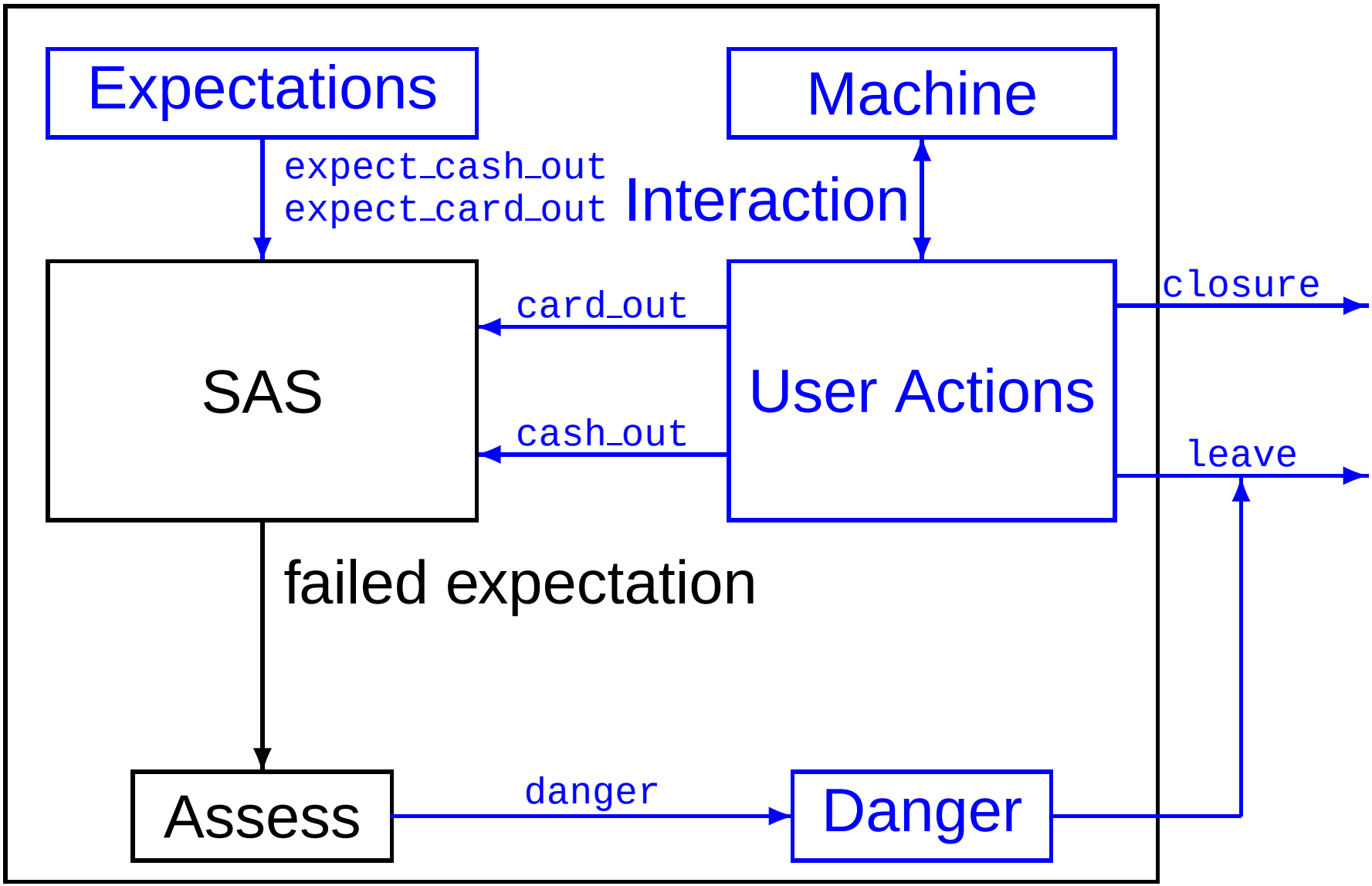
- expect cash_out before card_out

```
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
```

SAS in ATM: Expectations



Expectations Failure in ATM

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

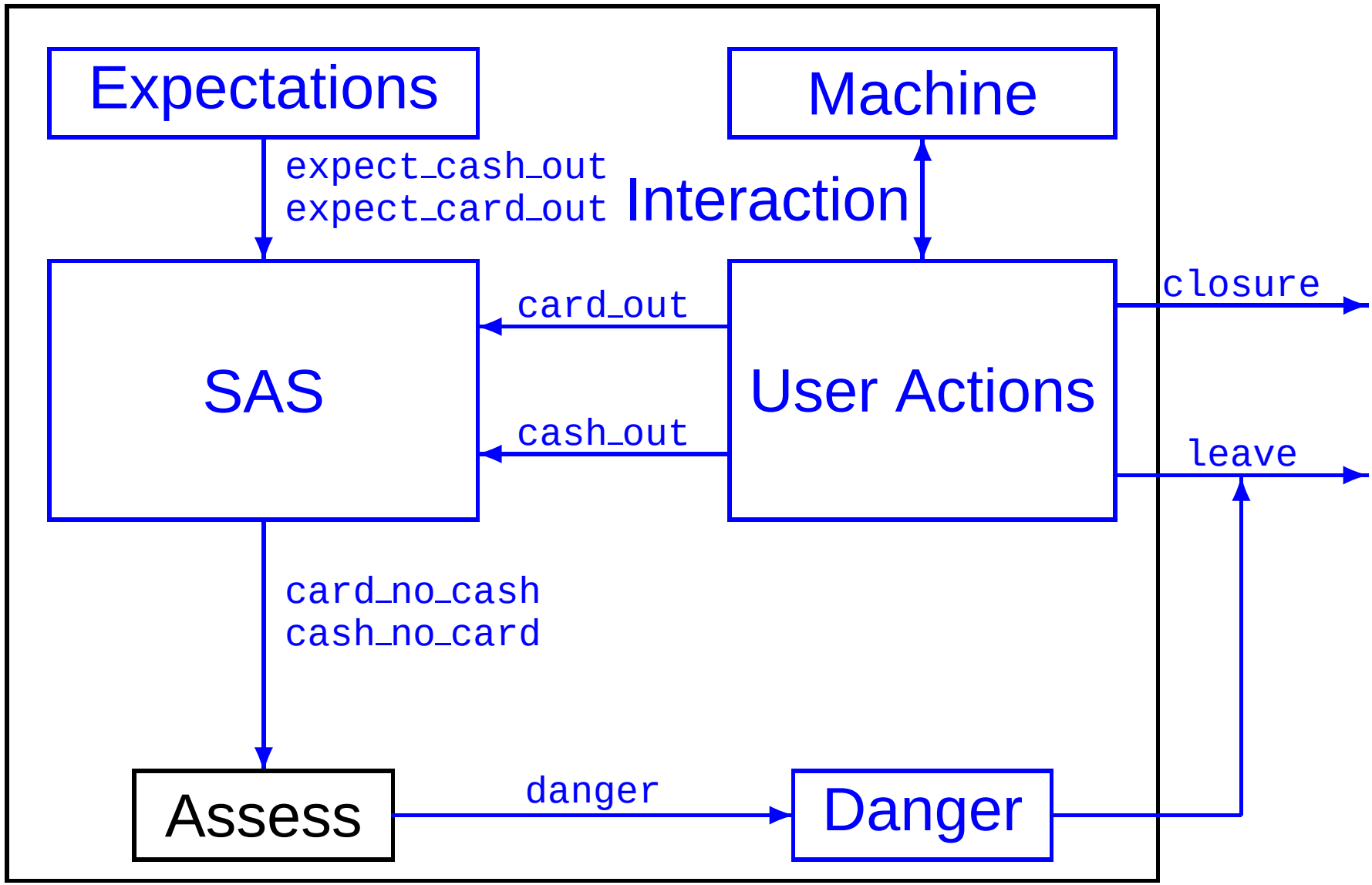
Card Expectations Failure

```
proc SAS = start_int -> Activation
         [] card_out -> SAS
         [] cash_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 )
```

SAS in ATM: Activation



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

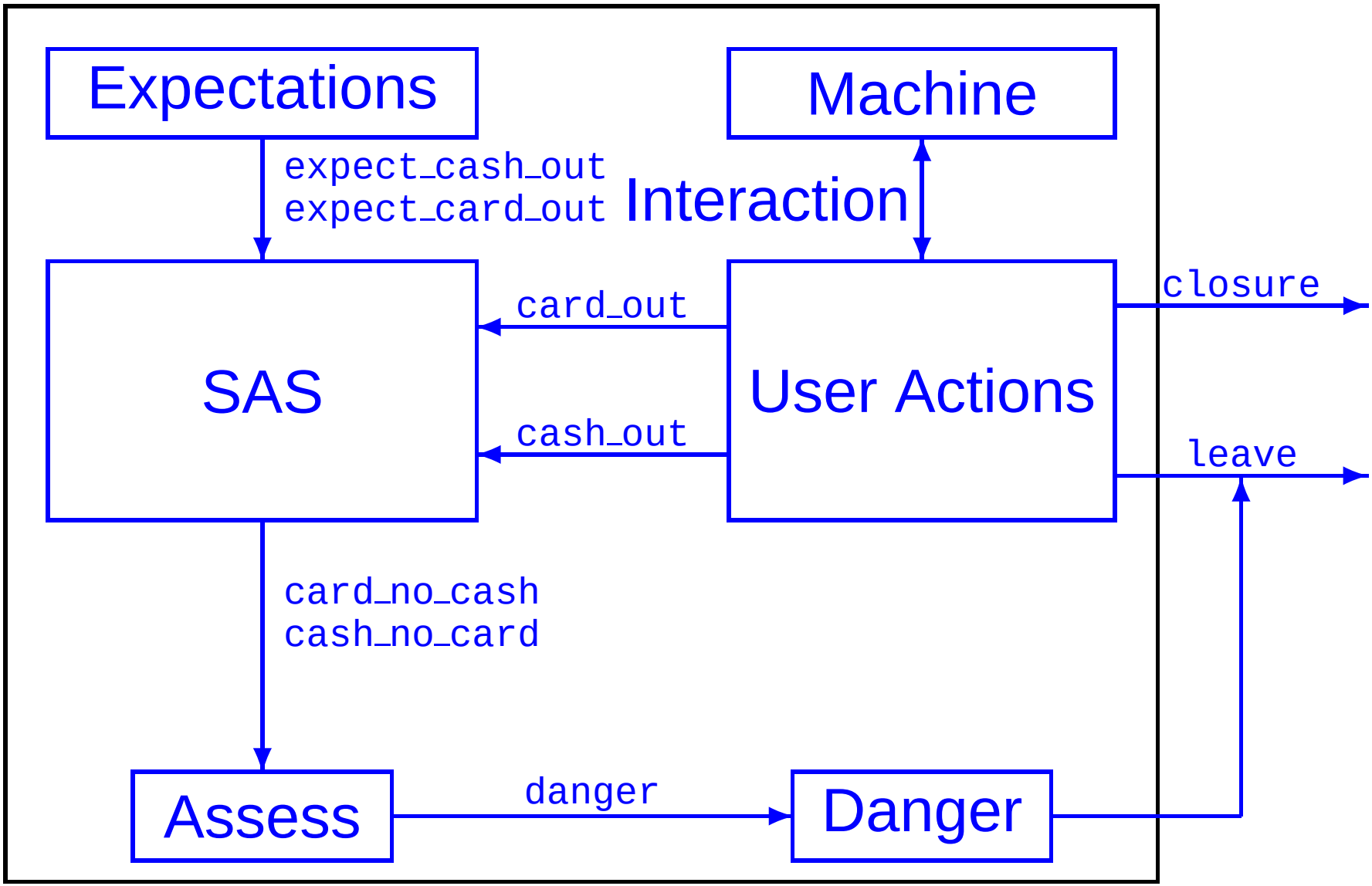
```
proc Assess =

 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
```

SAS in ATM: Assessment



Verifying Interactive Systems

- Started from an Informal Specification
- $\Longrightarrow$ Formal Model
  - Interface (Machine) $\Longleftarrow$ Implementation
  - Human (User) $=$ Cognitive Model
- $\Longrightarrow$ Formal Specification
  - unambiguous form of Task Specification
- Analysis
  - Formal Verification of the Interface in the presence of the Cognitive Model against the Specification

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

Formal HCI History

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]

References

[Huth and Ryan 04]

Michael Huth and Mark Ryan.
Logic in computer Science.
Cambridge University Press, 2nd Edition, 2004.

Textbook

One of the most complete general textbooks on the use of logics in computer science, it covers:

- Propositional Logic
- Predicate Logic
- Modal Logics
- Temporal Logics
- Formal Verification Approaches

[Parkin 02]

Alan J. Parkin.
Essential cognitive Psychology.
Psychology Press Press, 2002.

Textbook
Coincise but complete textbook on cognitive psychology

End