ve·ri·tas
[wey-ri-tahs; Eng. ver-i-tas, -tahs]
-noun Latin.
TRUTH.
WOPR
War Operation Plan Response

# RUNTIME VERIFICATION – AS AI GETS SMARTER, WHO WATCHES THE WATCHDOG? AND HOW?

HANNO HILDMANN

# SETTING THE SCENE

- Why it matters to us
- Where we come from
- What we have
- What we can do
- What we are doing ☺
- Discussion

# WHY IT MATTERS TO US

- We want to enable adaptivity during runtime
  - Systems that adapt to their environment and deliver consistent quality of service
  - Systems that assist the commanding officers even when they change their priorities
  - Systems that can overcome malfunctions by self-reconfiguration
  - … [your example here]

- The cost of failure is too high
  - Legally
  - Ethically
  - … factually

# WHY IT MATTERS TO US

- Today I want to talk about Runtime Verification

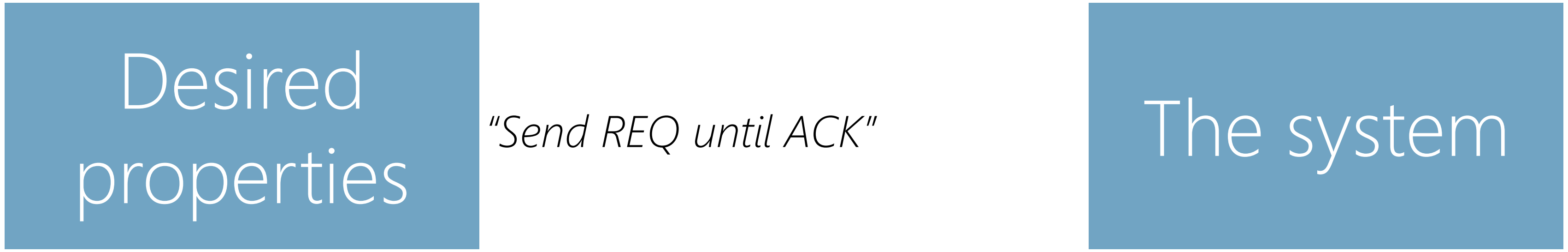
- This is not new, and the majority of my slides are about the current approach

- I will end by discussing the directions we are taking

- I (we) are hoping to engage in discussion and to hear your thoughts on
  - your need for Runtime Verification
  - ideas to achieve this
  - potential collaborations / brainstorming sessions / etc

# WHERE WE COME FROM

# WHERE WE COME FROM

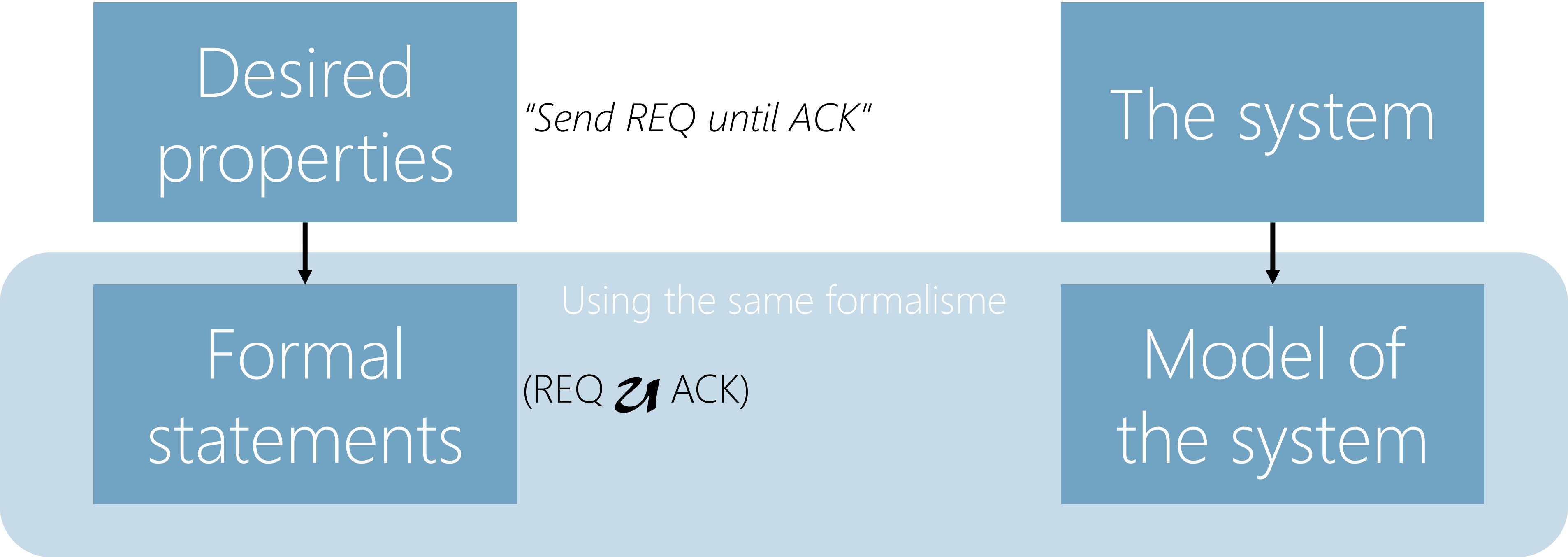


Desired properties

*"Send REQ until ACK"*

The system

# WHERE WE COME FROM



Desired properties

*"Send REQ until ACK"*

The system

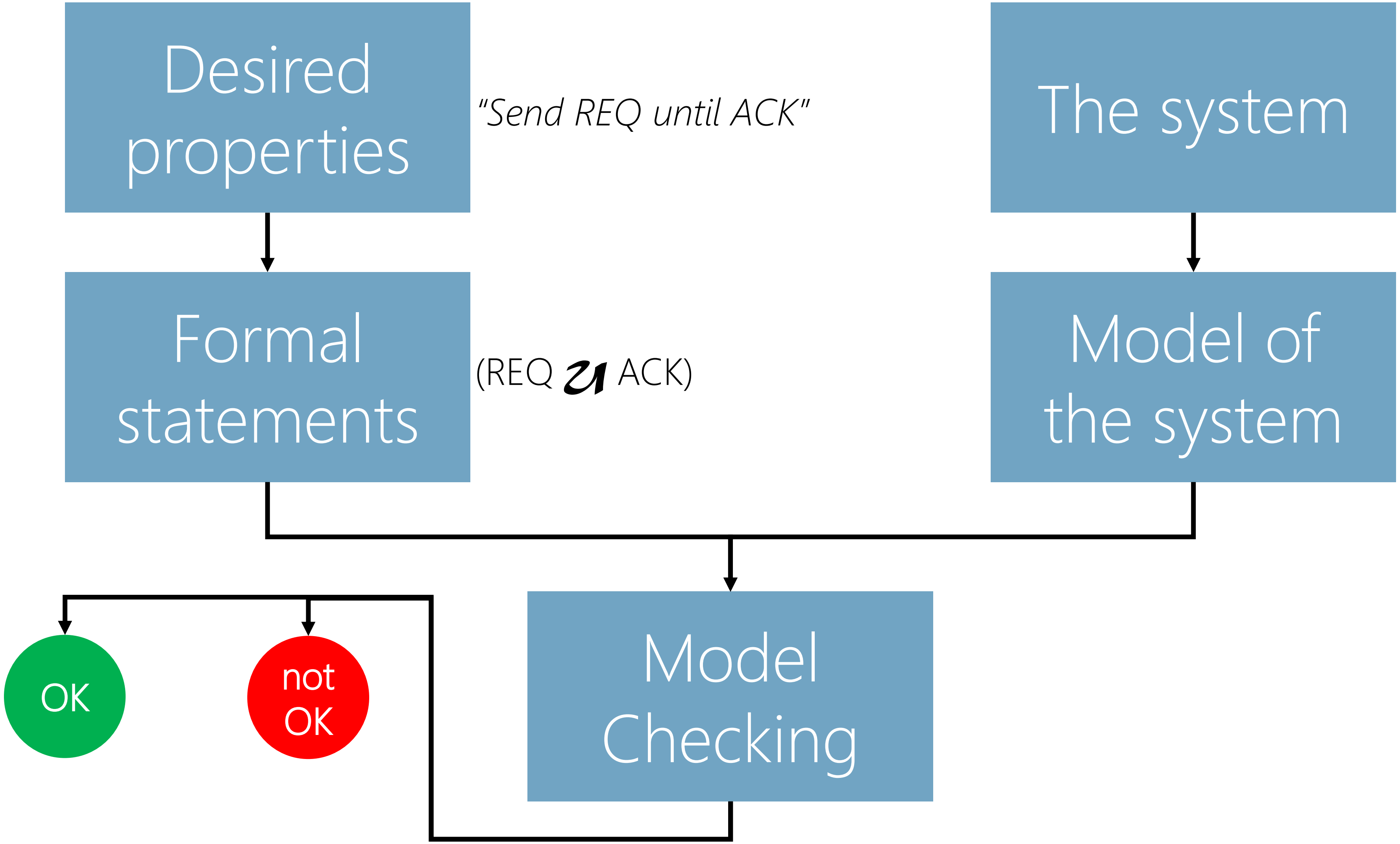
Using the same formalisme

Formal statements

(REQ $\mathcal{U}$ ACK)

Model of the system

# WHERE WE COME FROM



Desired properties

*"Send REQ until ACK"*

The system

Formal statements

$(REQ \; \mathcal{U} \; ACK)$

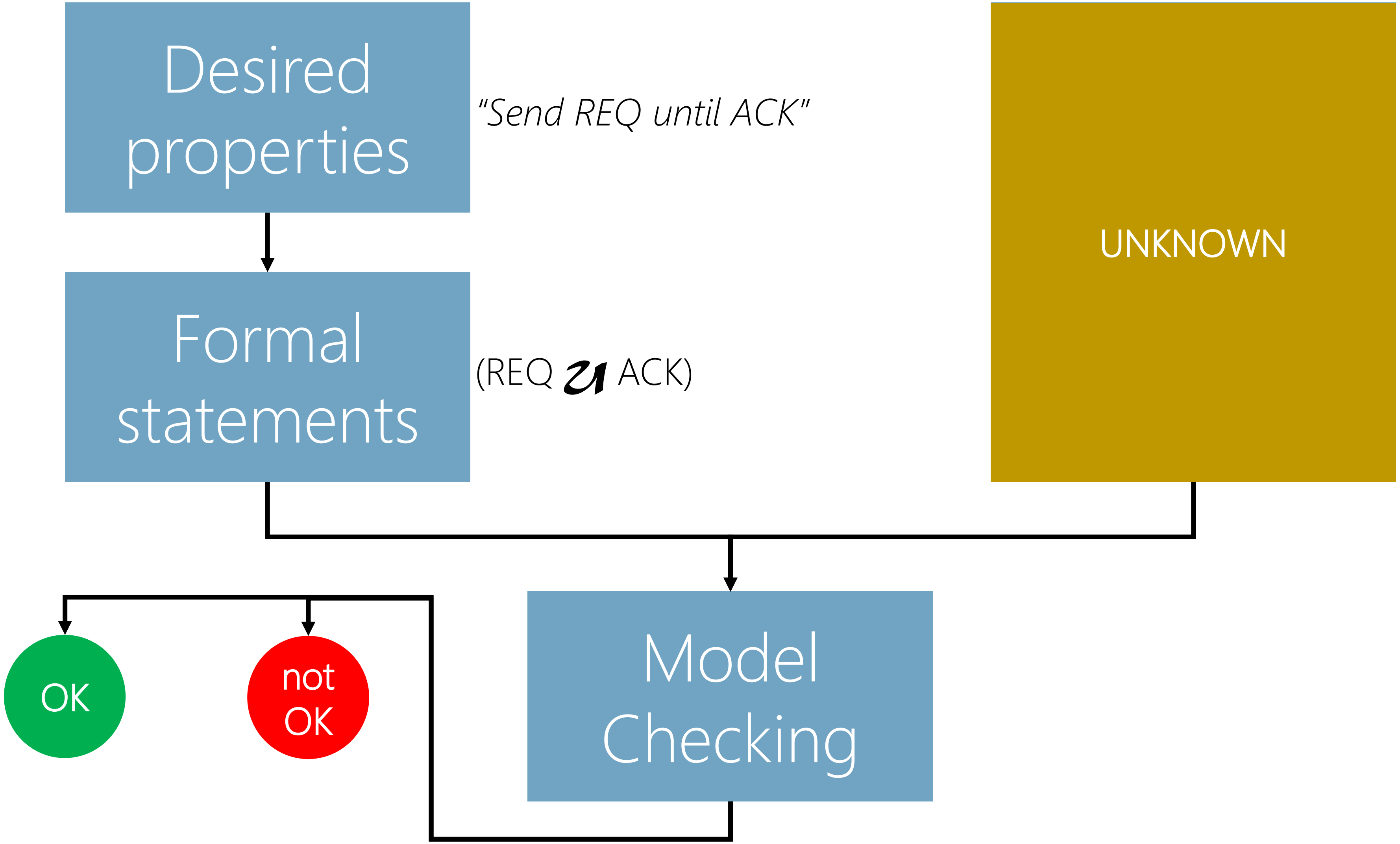
Model of the system

Model Checking

OK

not OK

# WHERE WE COME FROM



Desired properties

*"Send REQ until ACK"*

Formal statements

$(REQ \; \mathcal{U} \; ACK)$

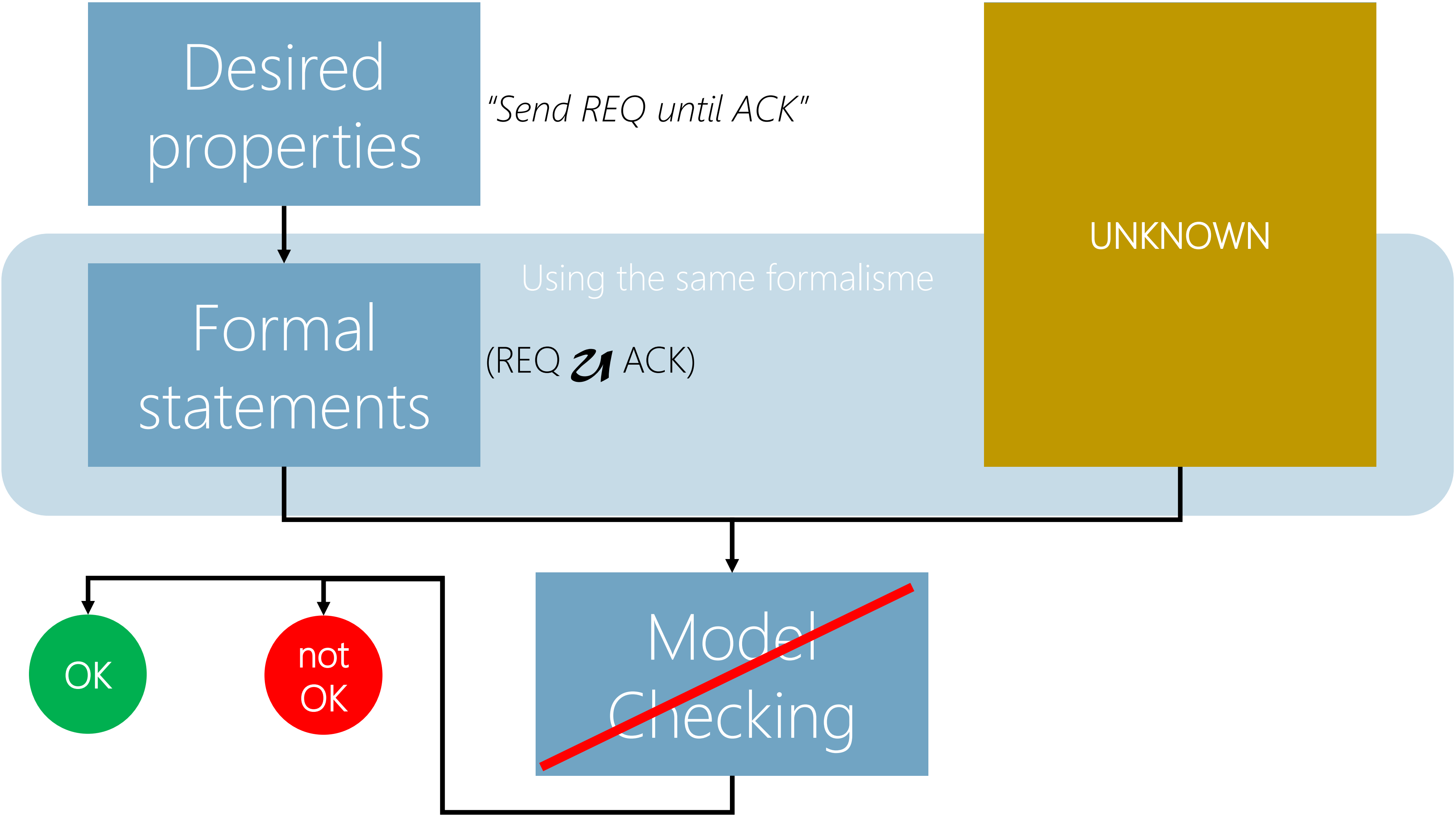
UNKNOWN

Model Checking

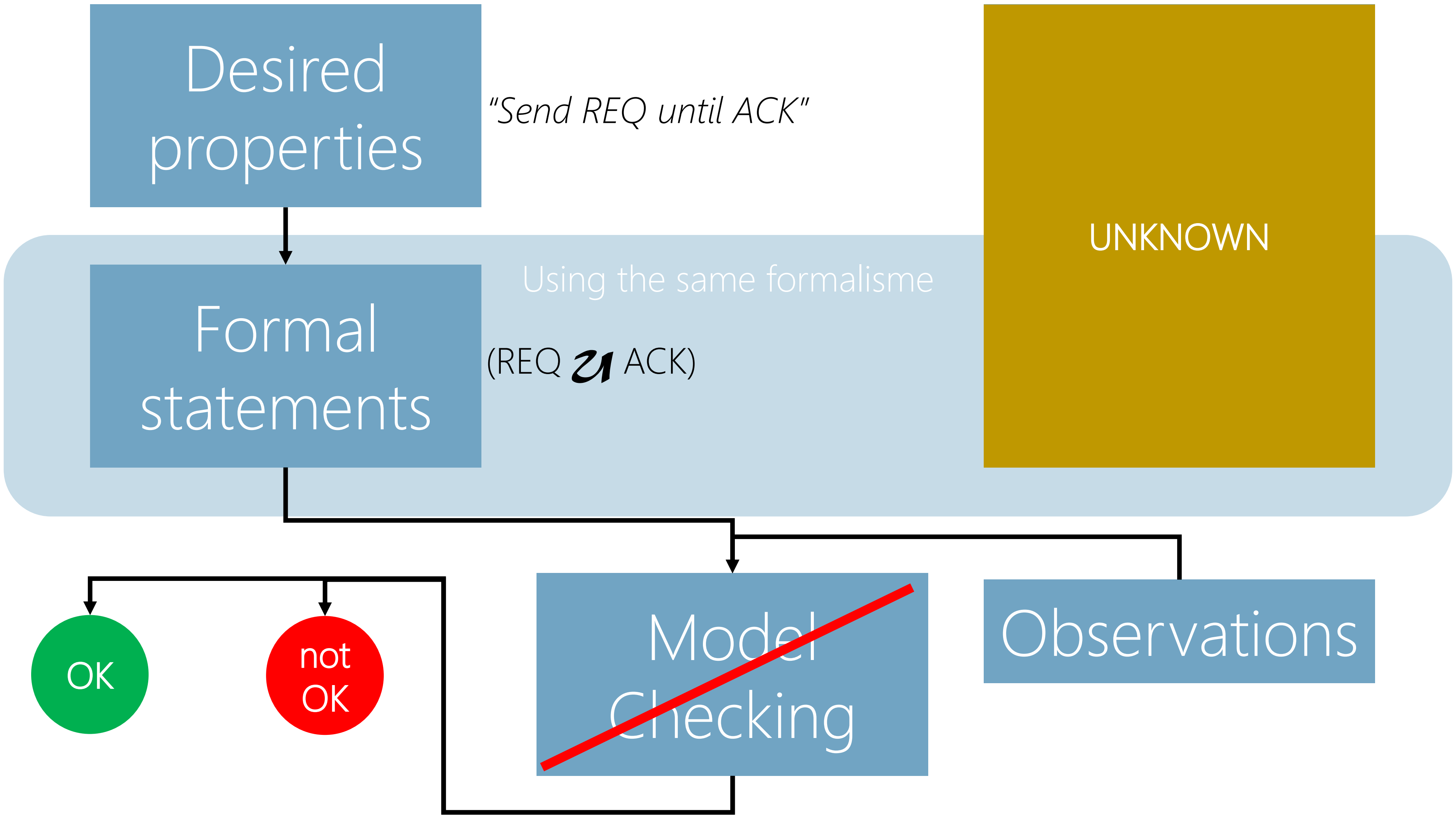
OK

not OK

# WHERE WE COME FROM



# WHERE WE COME FROM



Desired properties

*"Send REQ until ACK"*

Using the same formalisme

Formal statements

(REQ $\mathcal{U}$ ACK)

UNKNOWN

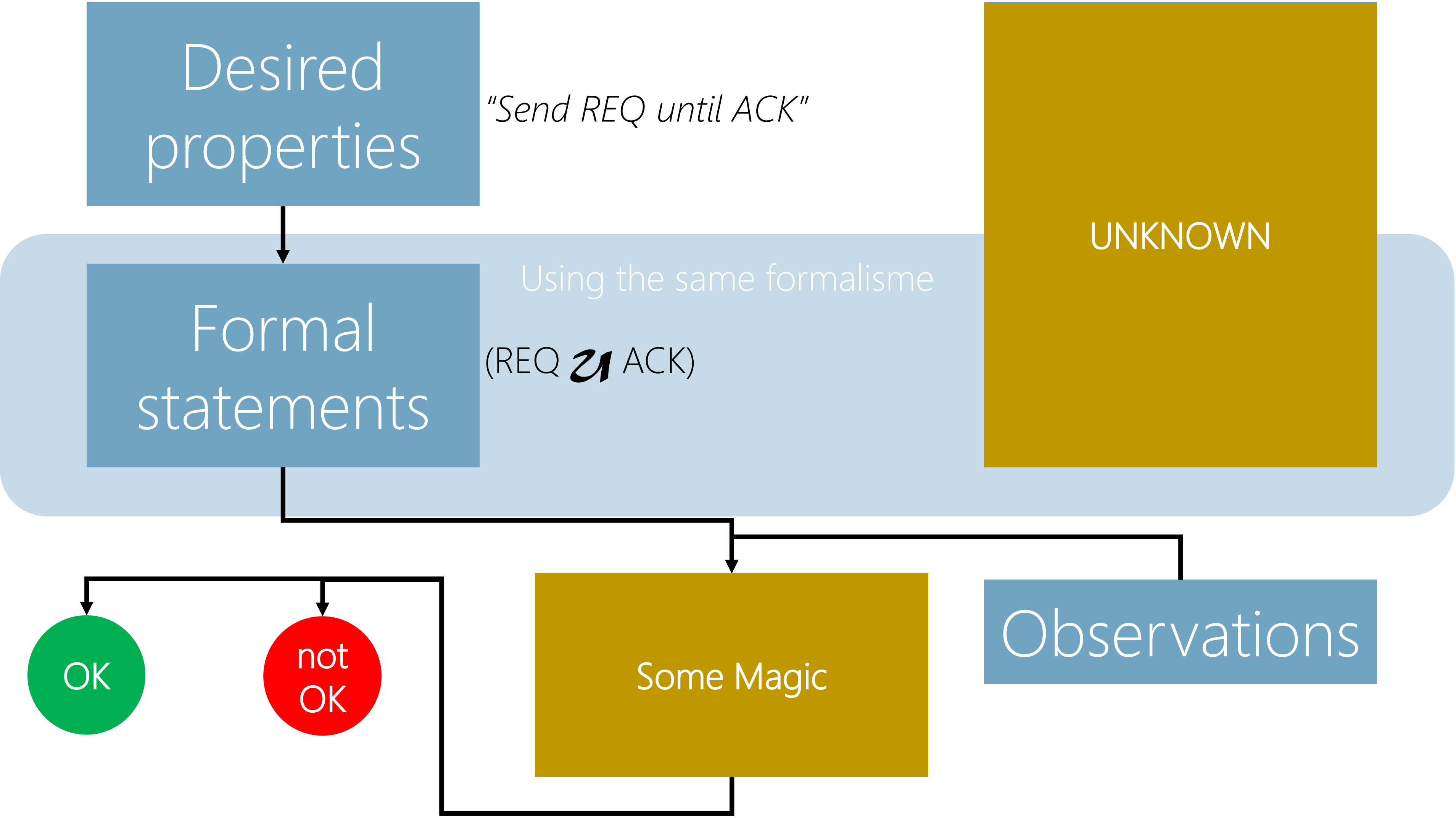
OK

not OK

~~Model Checking~~

Observations

# WHERE WE COME FROM



Desired properties

*"Send REQ until ACK"*

Formal statements

Using the same formalisme

(REQ $\mathcal{U}$ ACK)

UNKNOWN

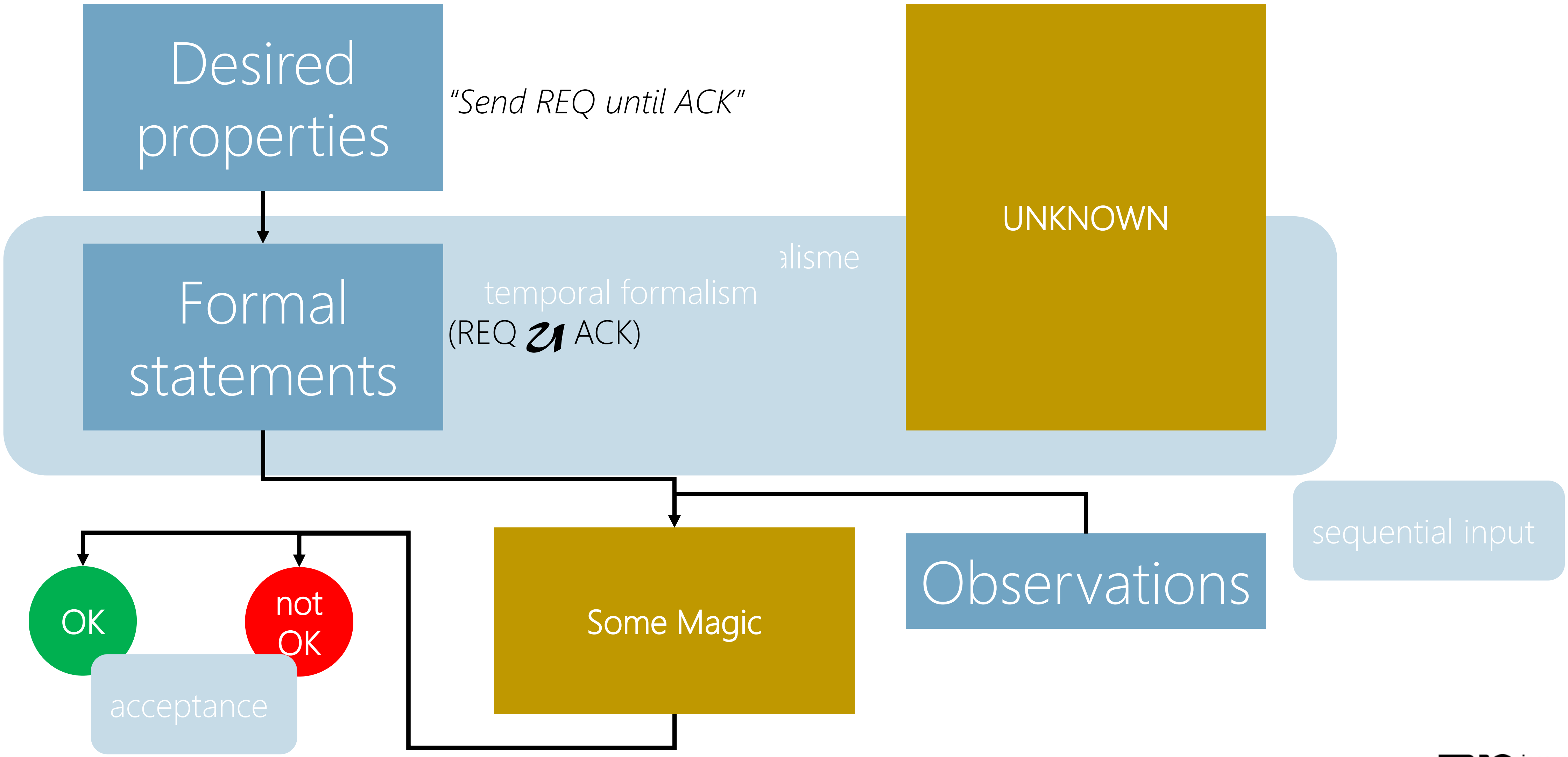
Some Magic

Observations

OK

not OK

# WHERE WE COME FROM



Desired properties

"Send REQ until ACK"

alisme

temporal formalism

$(REQ\ \mathcal{U}\ ACK)$

Formal statements

UNKNOWN

sequential input

Observations

Some Magic

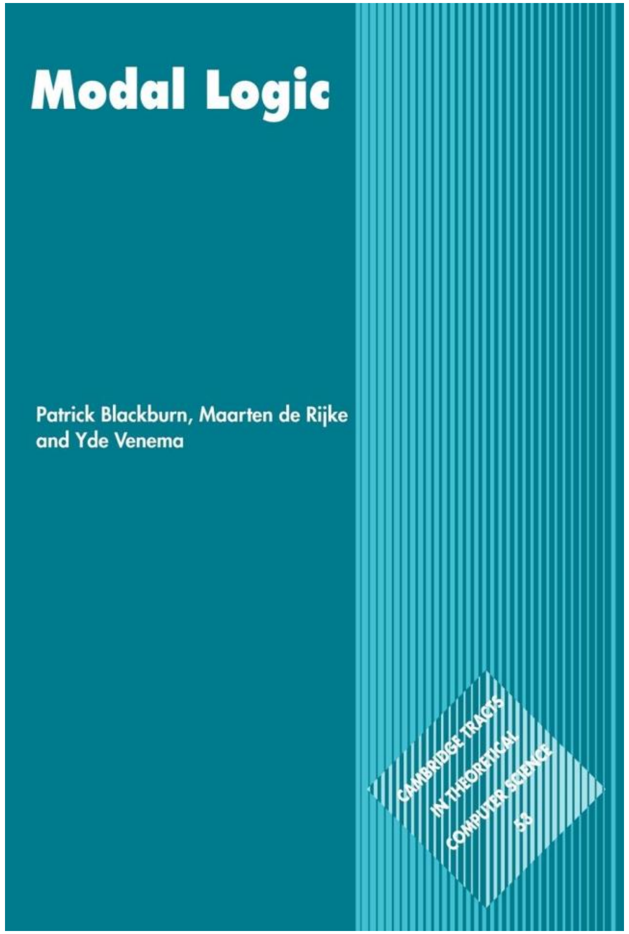
OK

not OK

acceptance

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)


Modal Logic
Patrick Blackburn, Maarten de Rijke and Yde Venema
CAMBRIDGE TRACTS IN THEORETICAL COMPUTER SCIENCE 53

Temporal Logics in Computer Science
Finite-State Systems
Stéphane Demri, Valentin Goranko and Martin Lange
Topology Via Logic
Steven Vickers
Logic and Computation
Interactive proof with Cambridge LCF
L.C.Paulson
Cambridge Tracts in Theoretical Computer Science 2
Epistemic Logic for AI and Computer Science
J.-J. Ch. Meyer and W. van der Hoek
Cambridge Tracts in Theoretical Computer Science 41
Extensions of First Order Logic
Maria Manzano
Cambridge Tracts in Theoretical Computer Science 19

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)

Propositional Logic

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)

Propositional Logic

p

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)

Propositional Logic

$p$

$\neg q$

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)

Propositional Logic

$p$

$\neg q$

$(p \wedge \neg q)$

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)

Propositional Logic

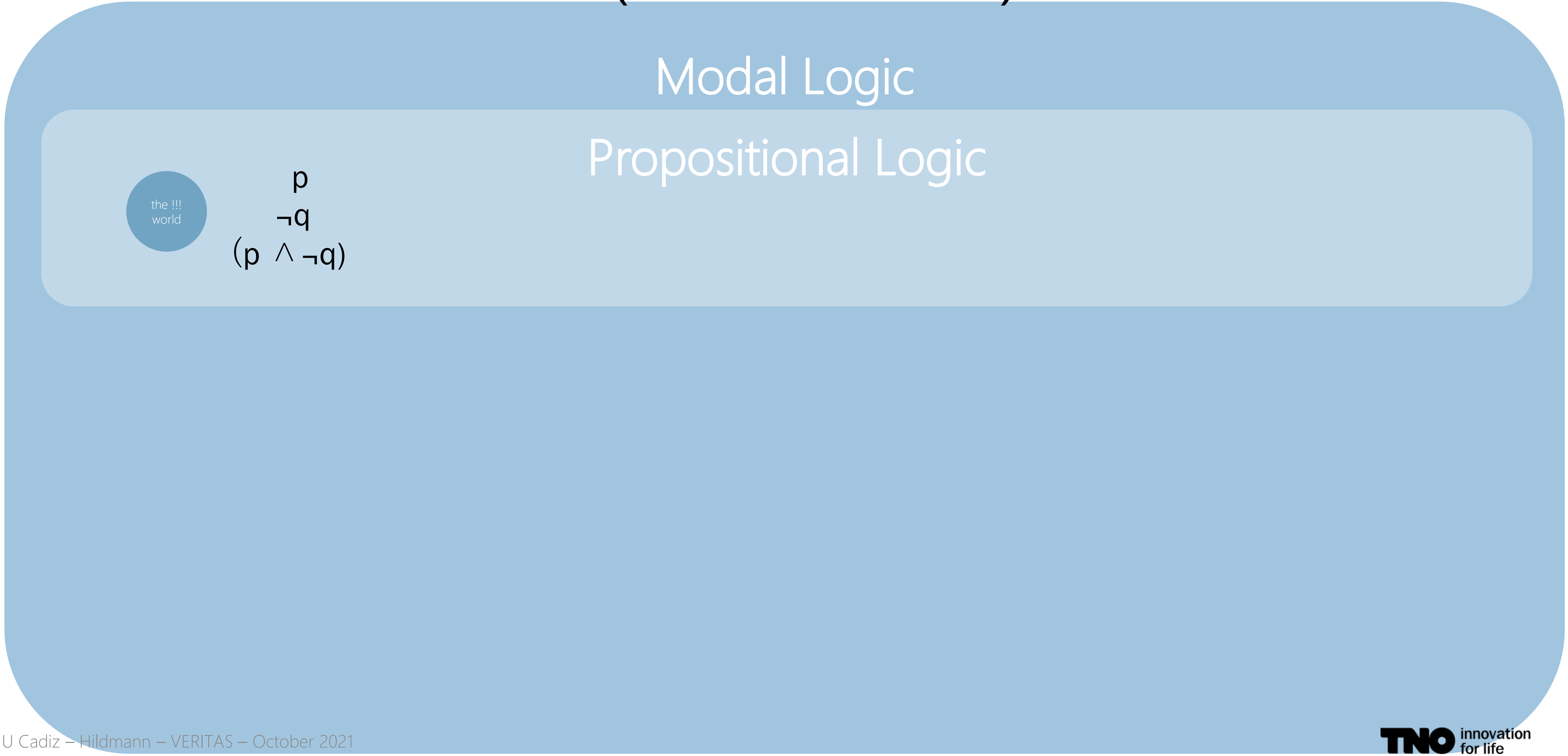
$p$

$\neg q$

$(p \wedge \neg q)$

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

the !!! world

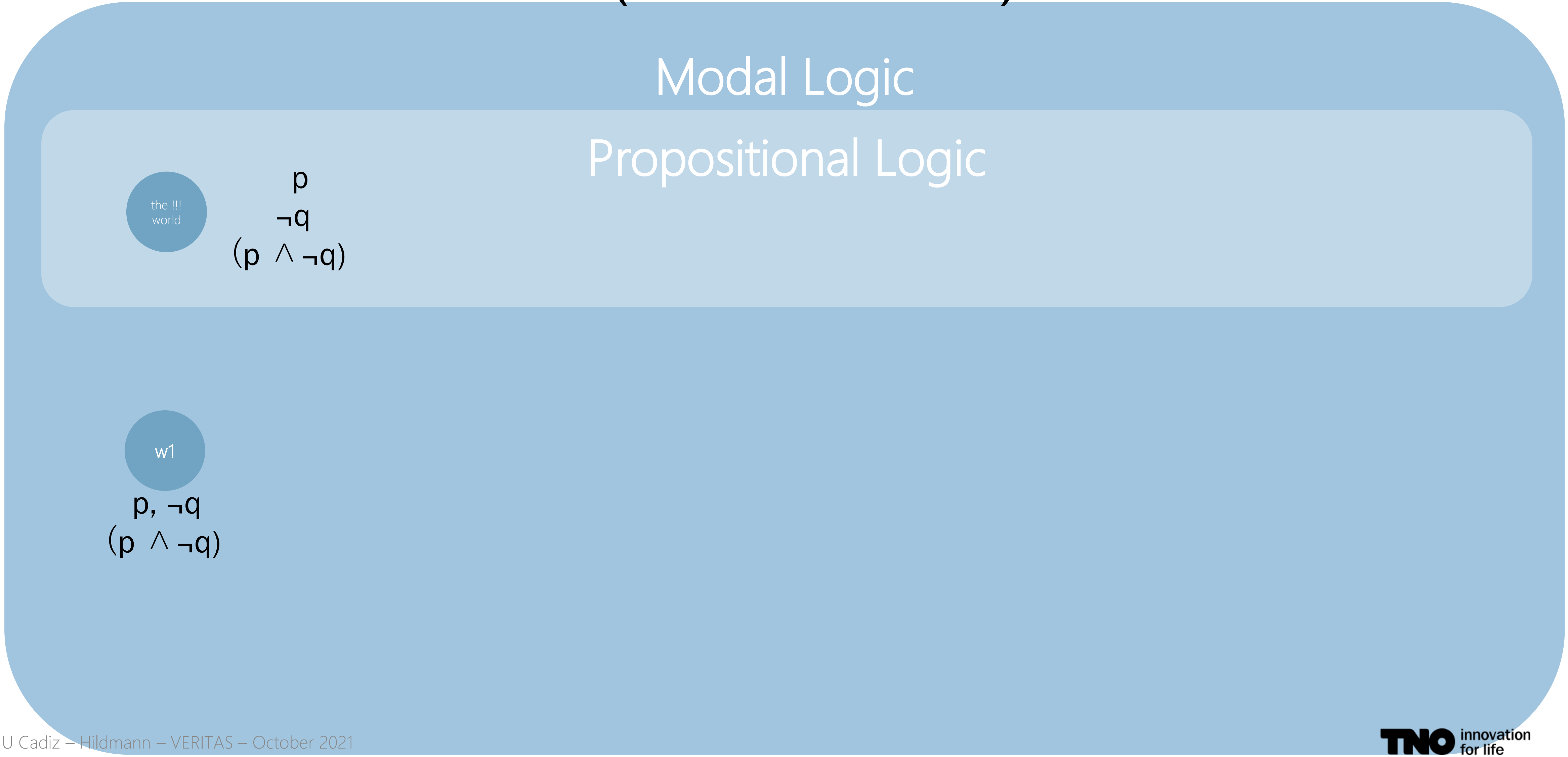
$p$

$\neg q$

$(p \wedge \neg q)$

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

the !!!
world

$p$

$\neg q$

$(p \land \neg q)$

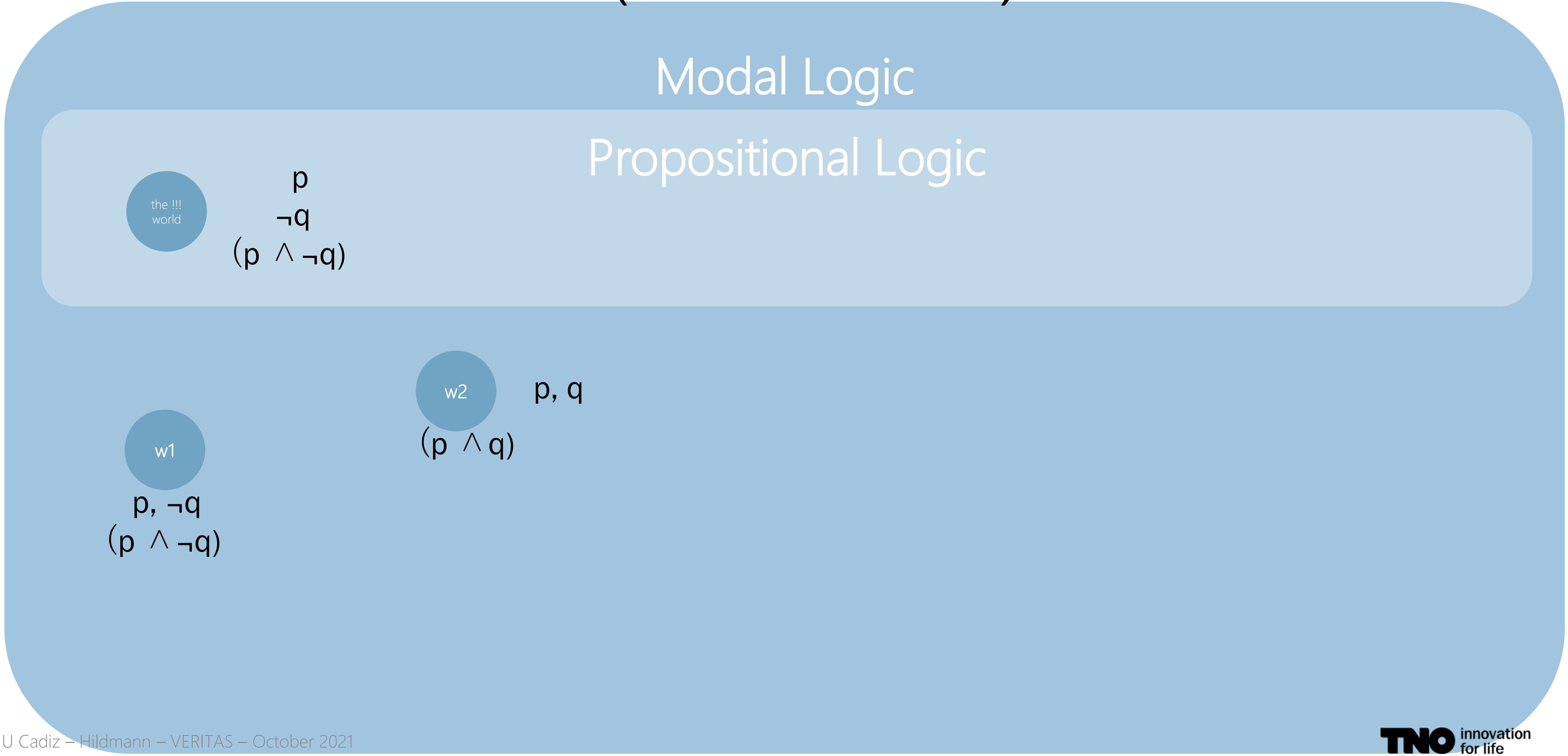
w1

$p, \neg q$

$(p \land \neg q)$

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

the !!! world

p

¬q

(p ∧ ¬q)

w2

p, q

(p ∧ q)

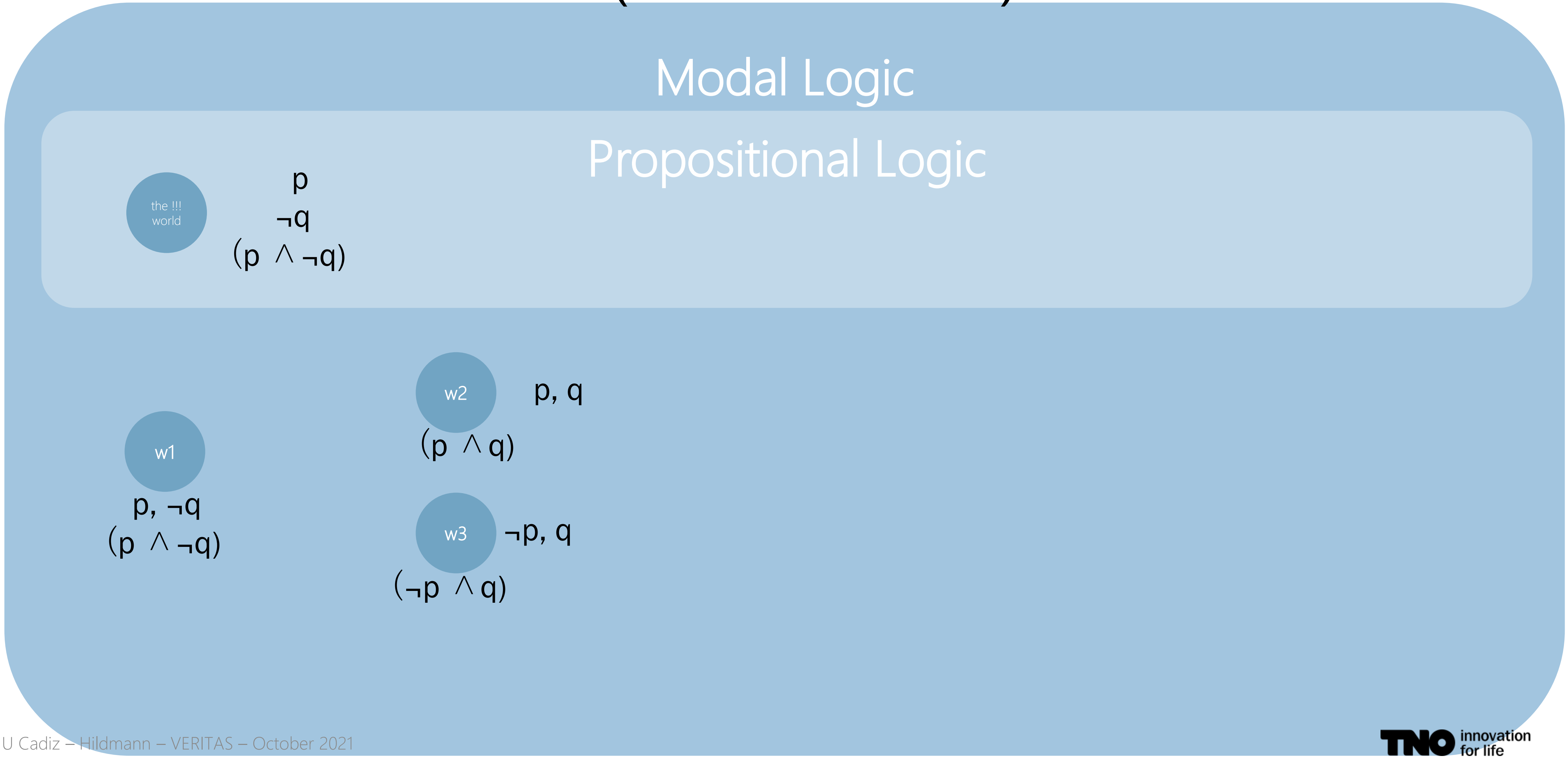
w1

p, ¬q

(p ∧ ¬q)

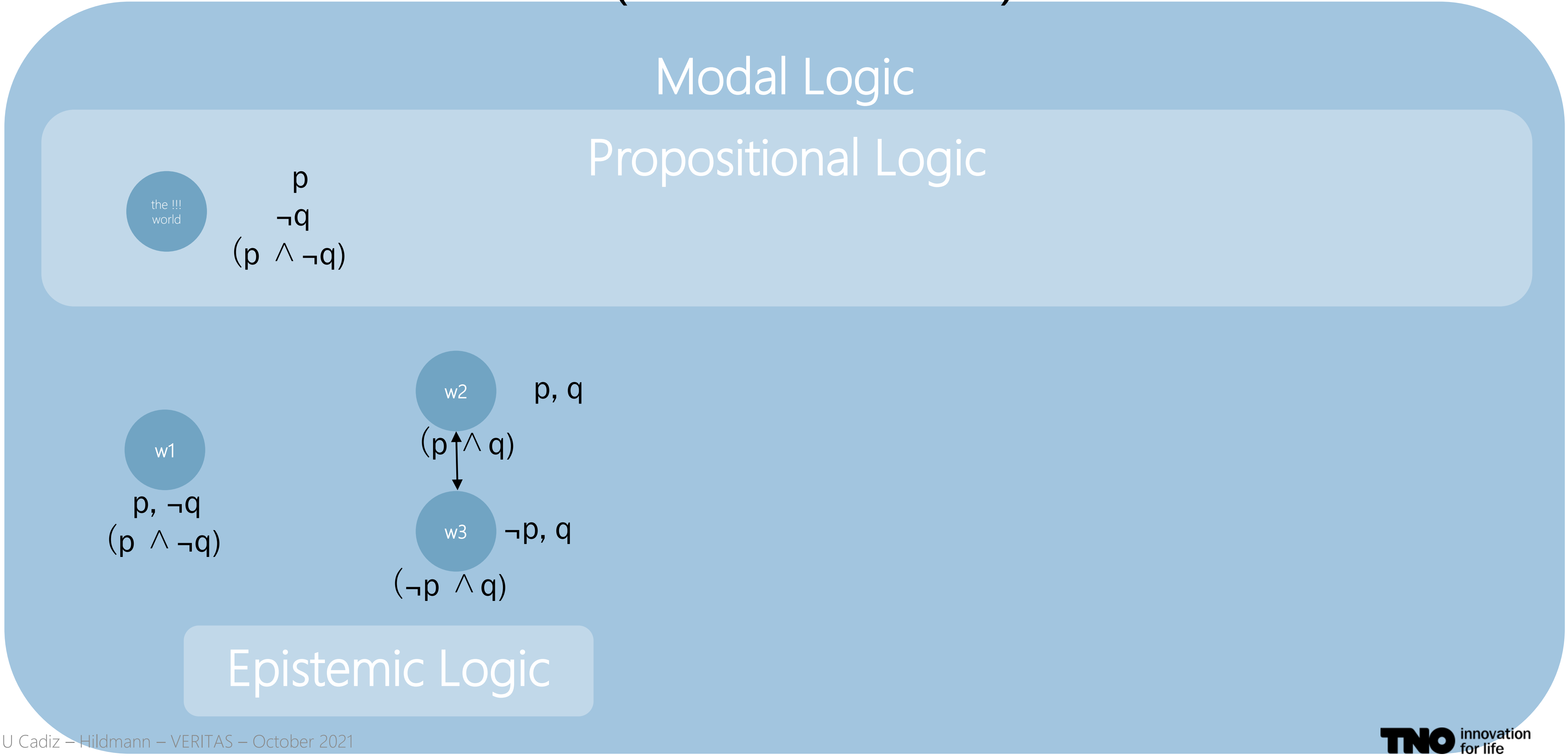
# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

the !!! world

$p$

$\neg q$

$(p \wedge \neg q)$

w1

$p, \neg q$

$(p \wedge \neg q)$

w2

$p, q$

$(p \wedge q)$

w3

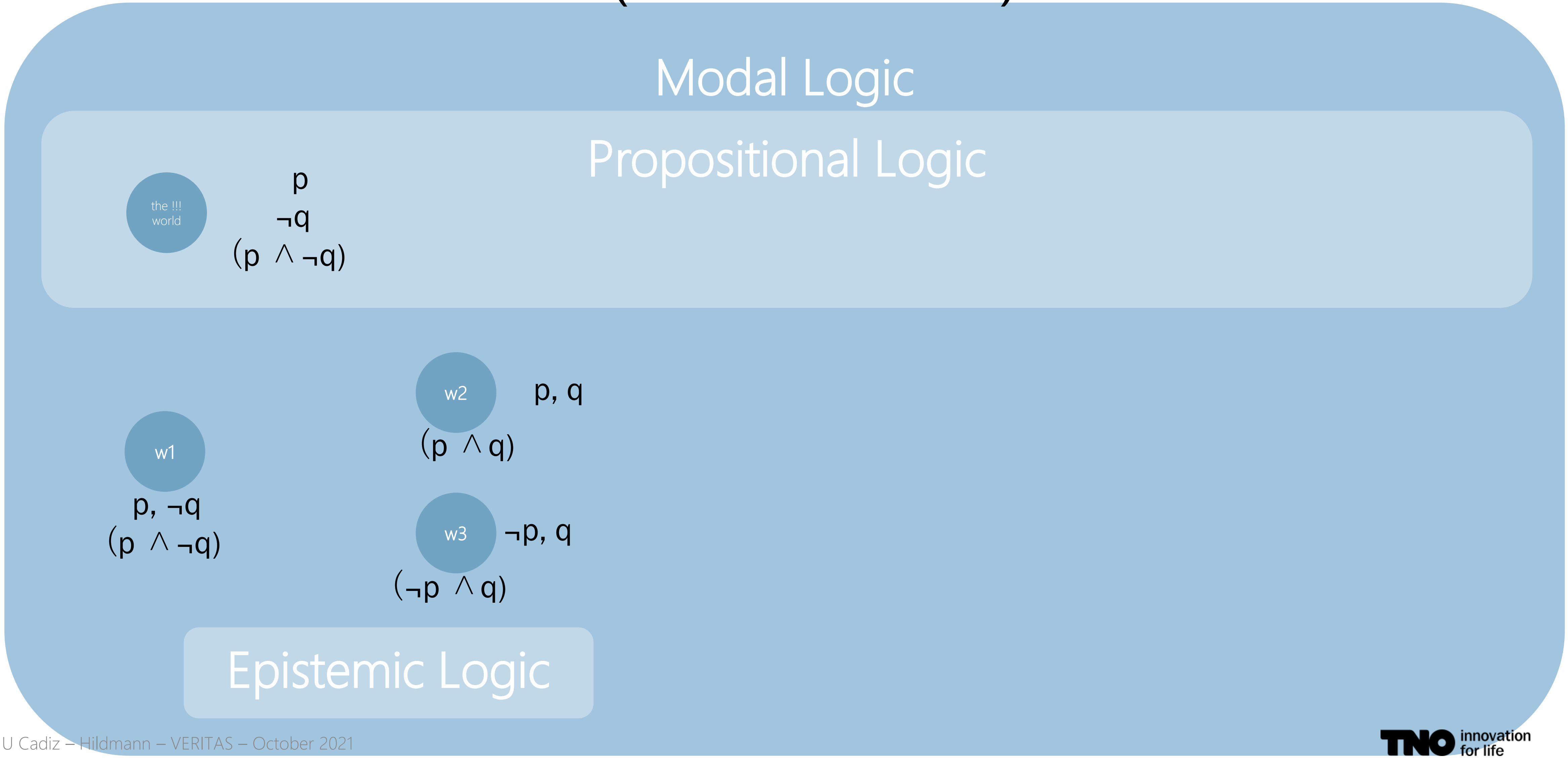
$\neg p, q$

$(\neg p \wedge q)$

Epistemic Logic

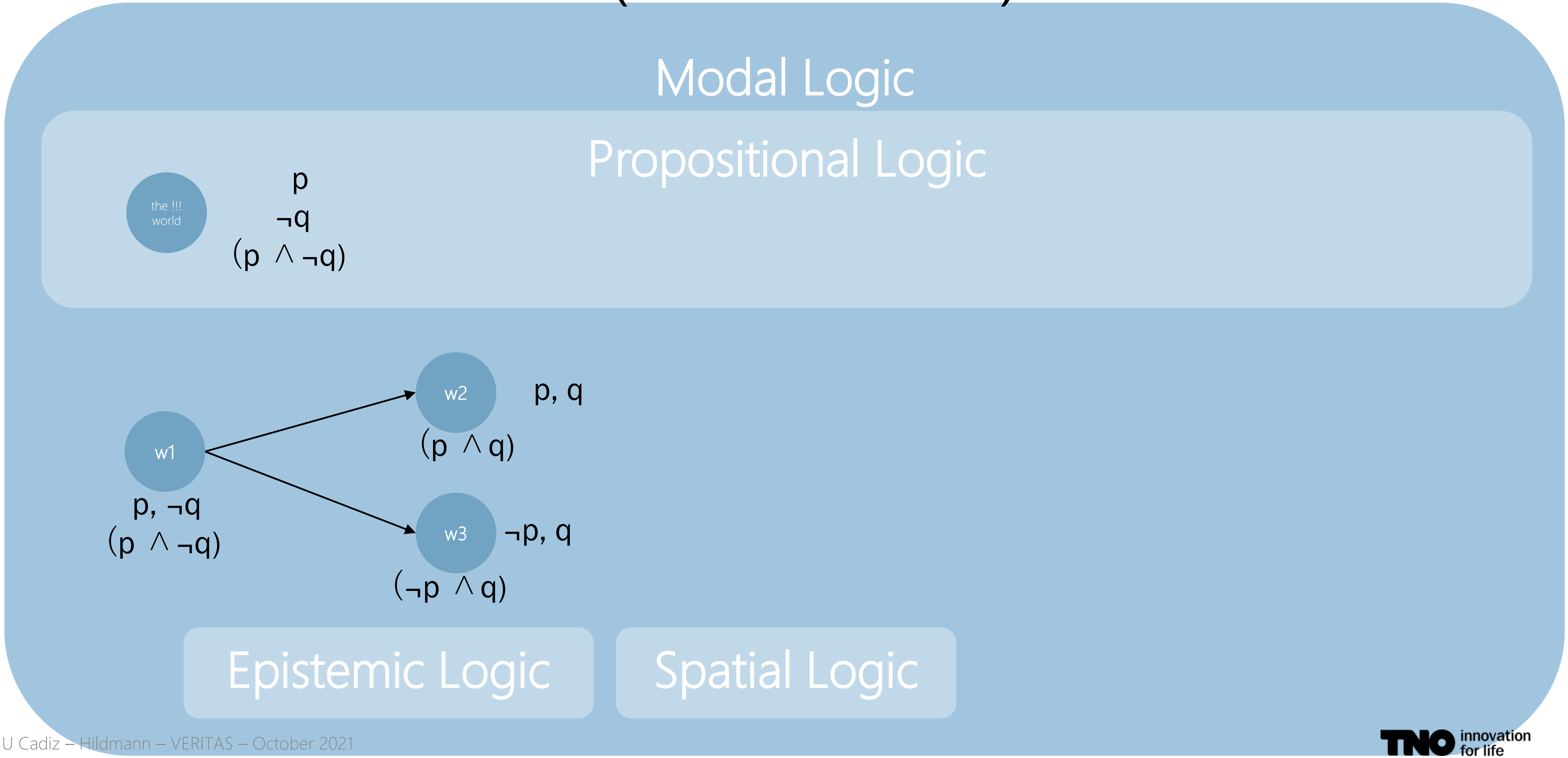
# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

the !!! world

$p$

$\neg q$

$(p \wedge \neg q)$

w1

$p, \neg q$

$(p \wedge \neg q)$

w2

$p, q$

$(p \wedge q)$

w3

$\neg p, q$

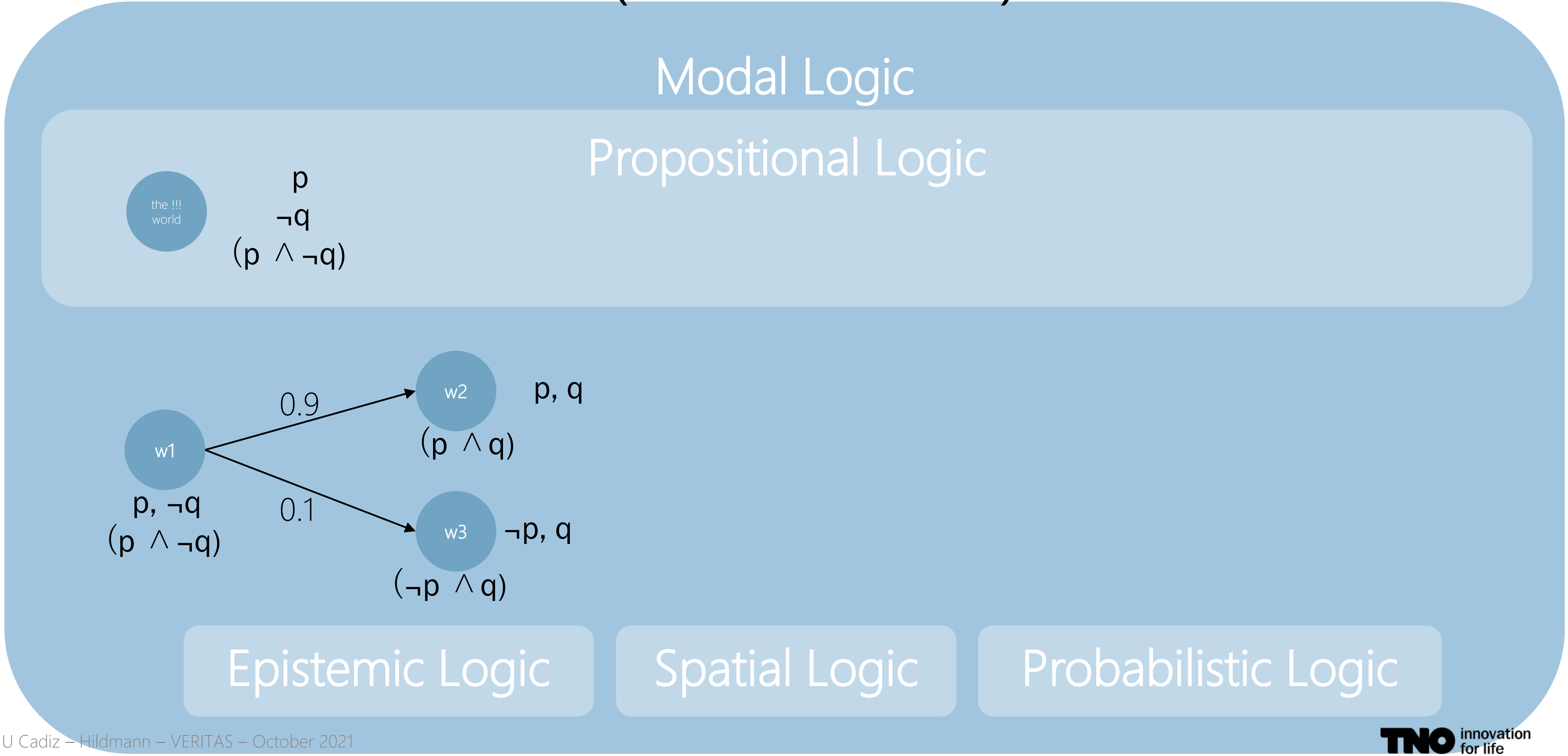
$(\neg p \wedge q)$

Epistemic Logic

Spatial Logic

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

the !!! world

$p$

$\neg q$

$(p \wedge \neg q)$

w1

$p, \neg q$

$(p \wedge \neg q)$

0.9

w2

$p, q$

$(p \wedge q)$

0.1

w3

$\neg p, q$

$(\neg p \wedge q)$

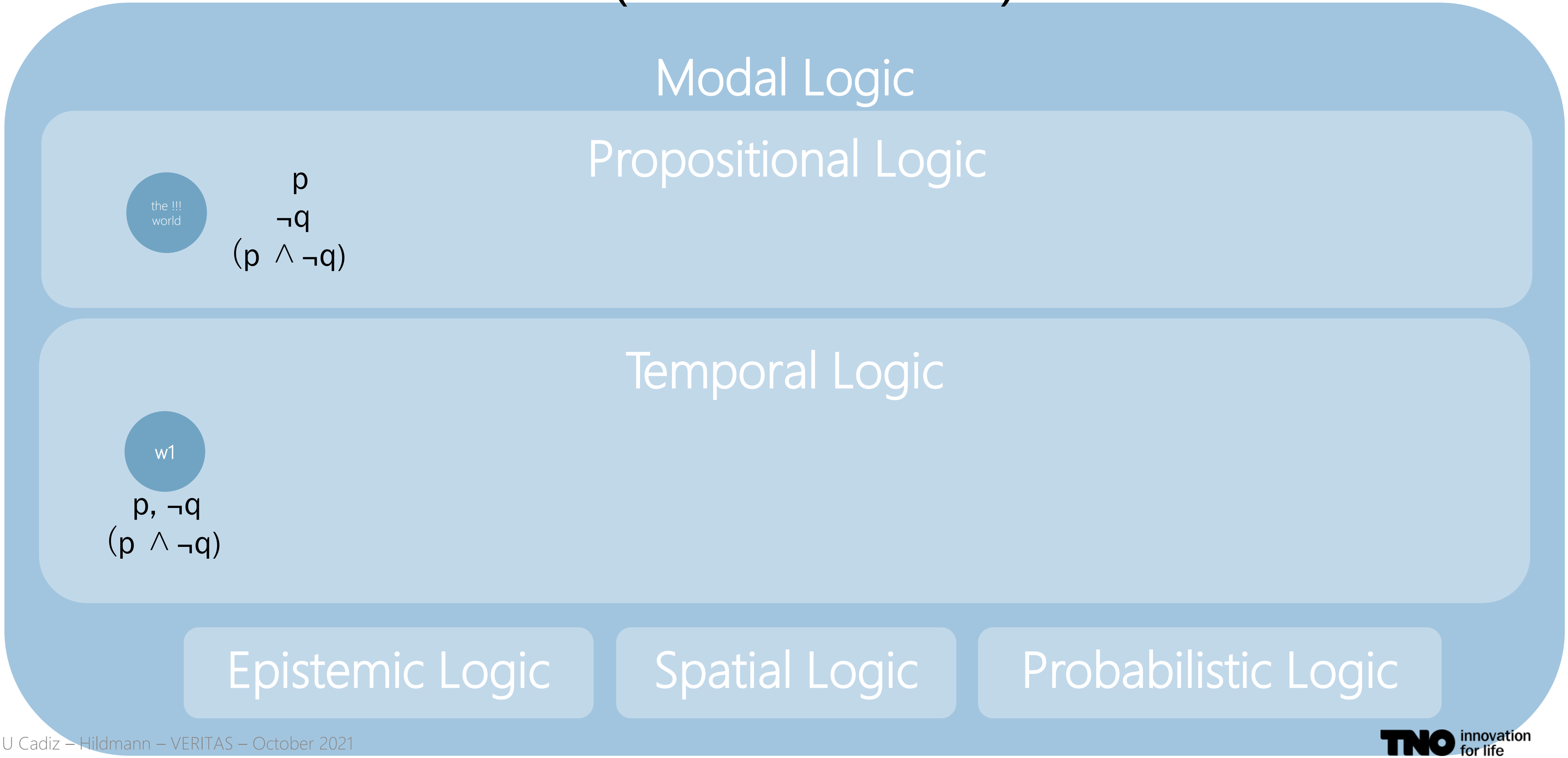
Epistemic Logic

Spatial Logic

Probabilistic Logic

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

the !!!
world

$p$

$\neg q$

$(p \wedge \neg q)$

Temporal Logic

w1

$p, \neg q$

$(p \wedge \neg q)$

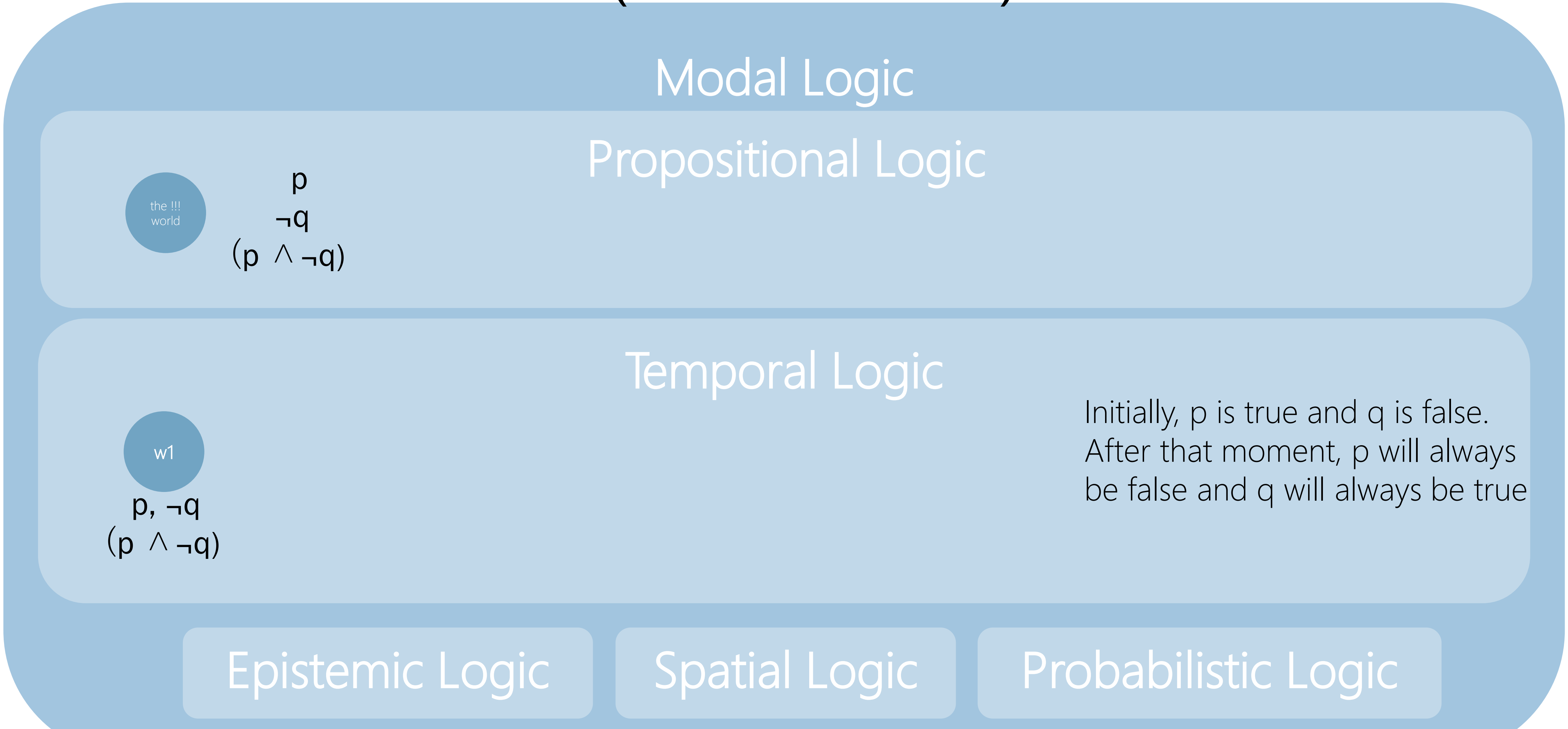
Epistemic Logic

Spatial Logic

Probabilistic Logic

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

the !!! world

$p$

$\neg q$

$(p \wedge \neg q)$

Temporal Logic

w1

$p, \neg q$

$(p \wedge \neg q)$

Initially, p is true and q is false. After that moment, p will always be false and q will always be true

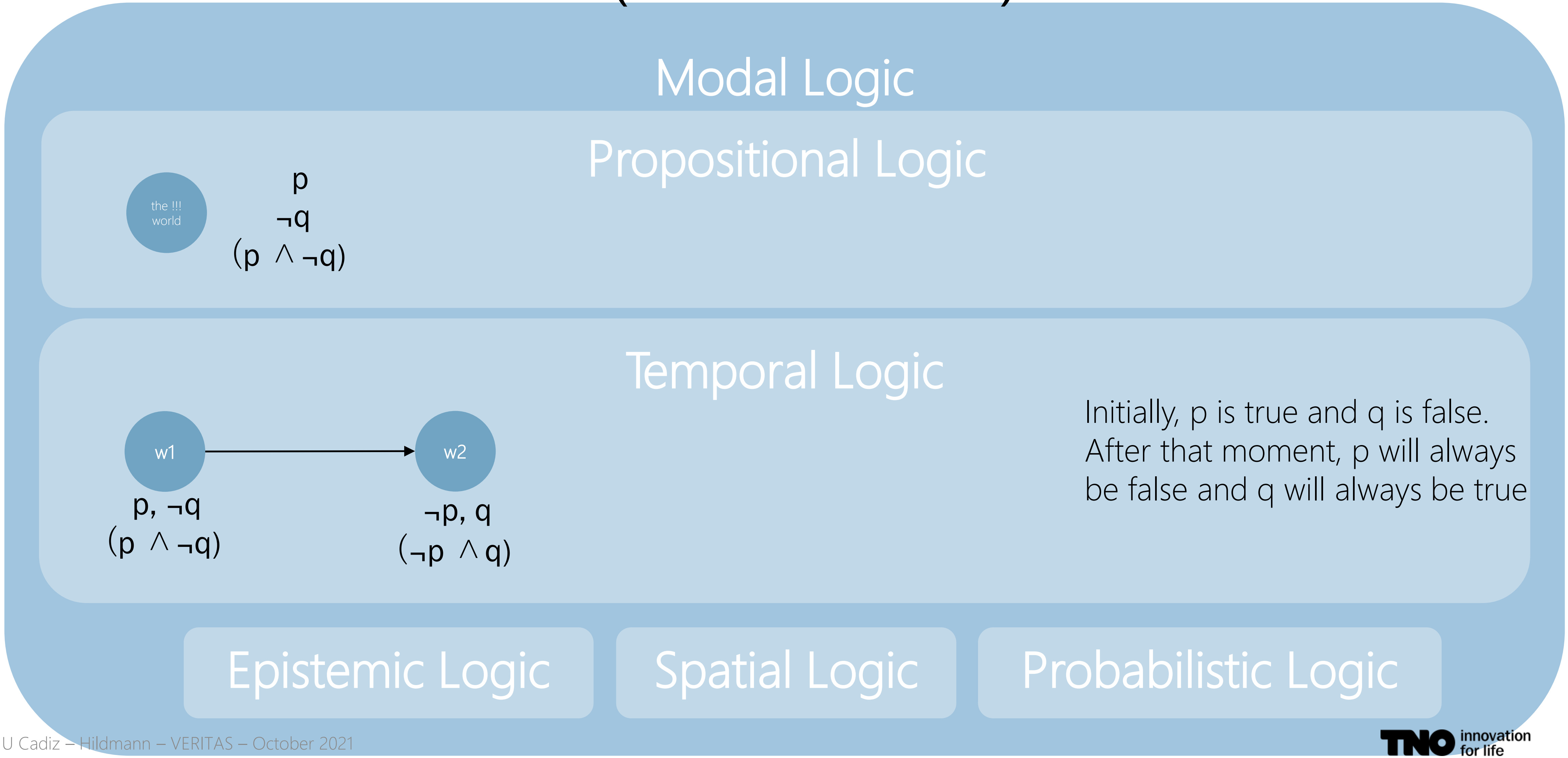
Epistemic Logic

Spatial Logic

Probabilistic Logic

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

$p$

$\neg q$

$(p \land \neg q)$

Temporal Logic

$p, \neg q$

$(p \land \neg q)$

$\neg p, q$

$(\neg p \land q)$

Initially, p is true and q is false. After that moment, p will always be false and q will always be true

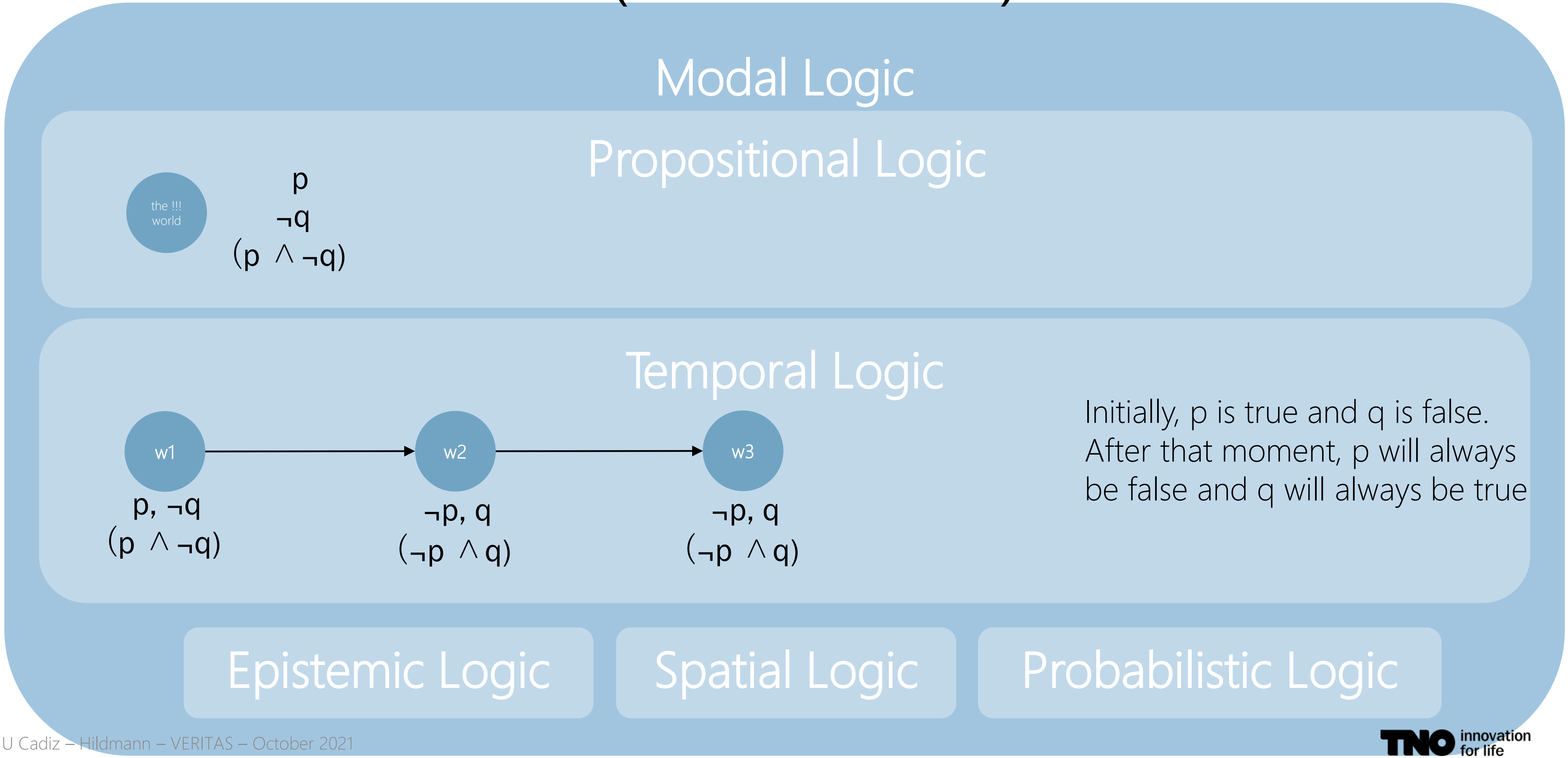
Epistemic Logic

Spatial Logic

Probabilistic Logic

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

the !!! world

$p$

$\neg q$

$(p \wedge \neg q)$

Temporal Logic

w1

$p, \neg q$

$(p \wedge \neg q)$

w2

$\neg p, q$

$(\neg p \wedge q)$

w3

$\neg p, q$

$(\neg p \wedge q)$

Initially, p is true and q is false.
After that moment, p will always
be false and q will always be true

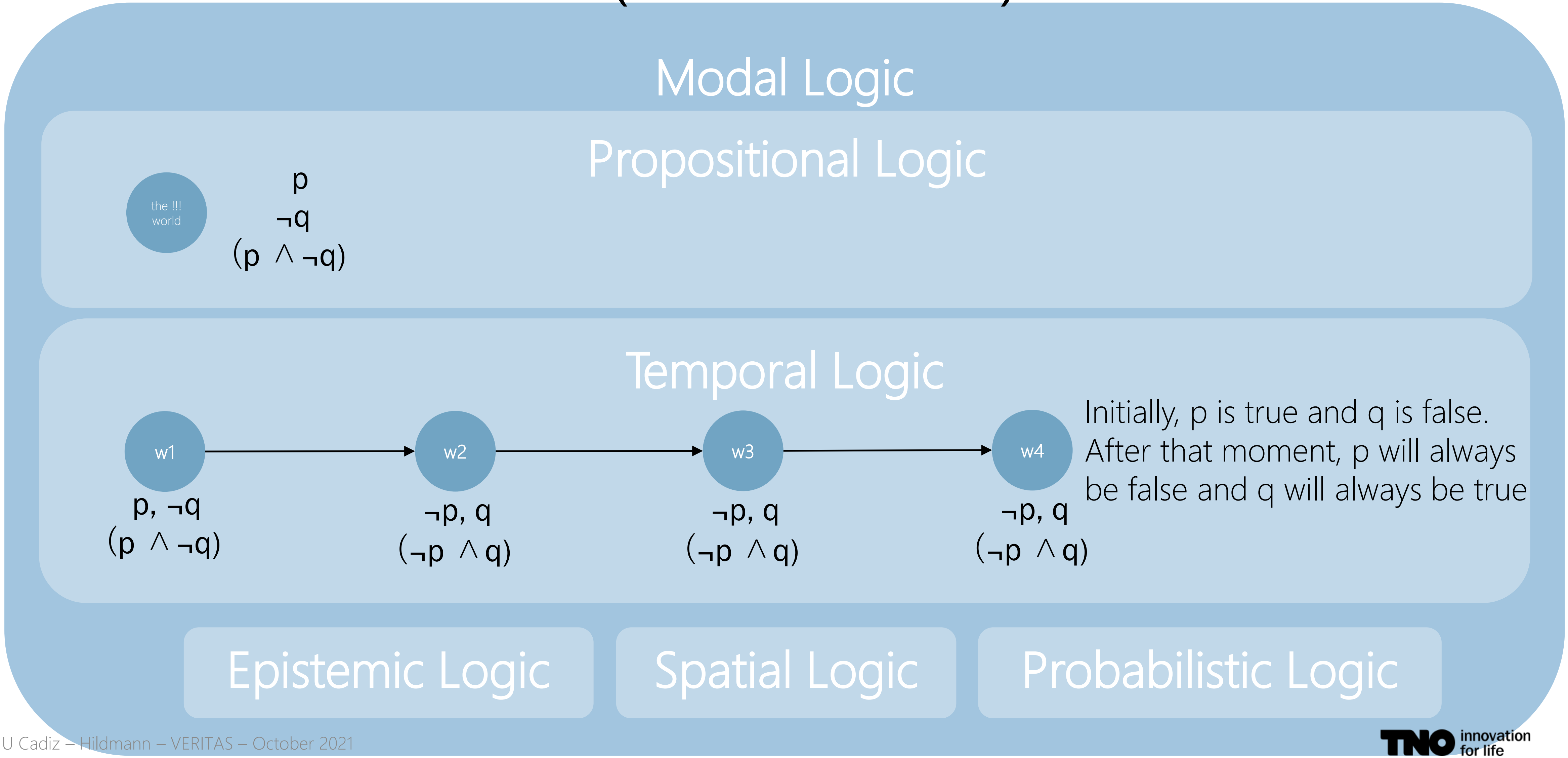
Epistemic Logic

Spatial Logic

Probabilistic Logic

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

$p$

$\neg q$

$(p \wedge \neg q)$

Temporal Logic

w1: $p, \neg q$ $(p \wedge \neg q)$

w2: $\neg p, q$ $(\neg p \wedge q)$

w3: $\neg p, q$ $(\neg p \wedge q)$

w4: $\neg p, q$ $(\neg p \wedge q)$

Initially, p is true and q is false. After that moment, p will always be false and q will always be true

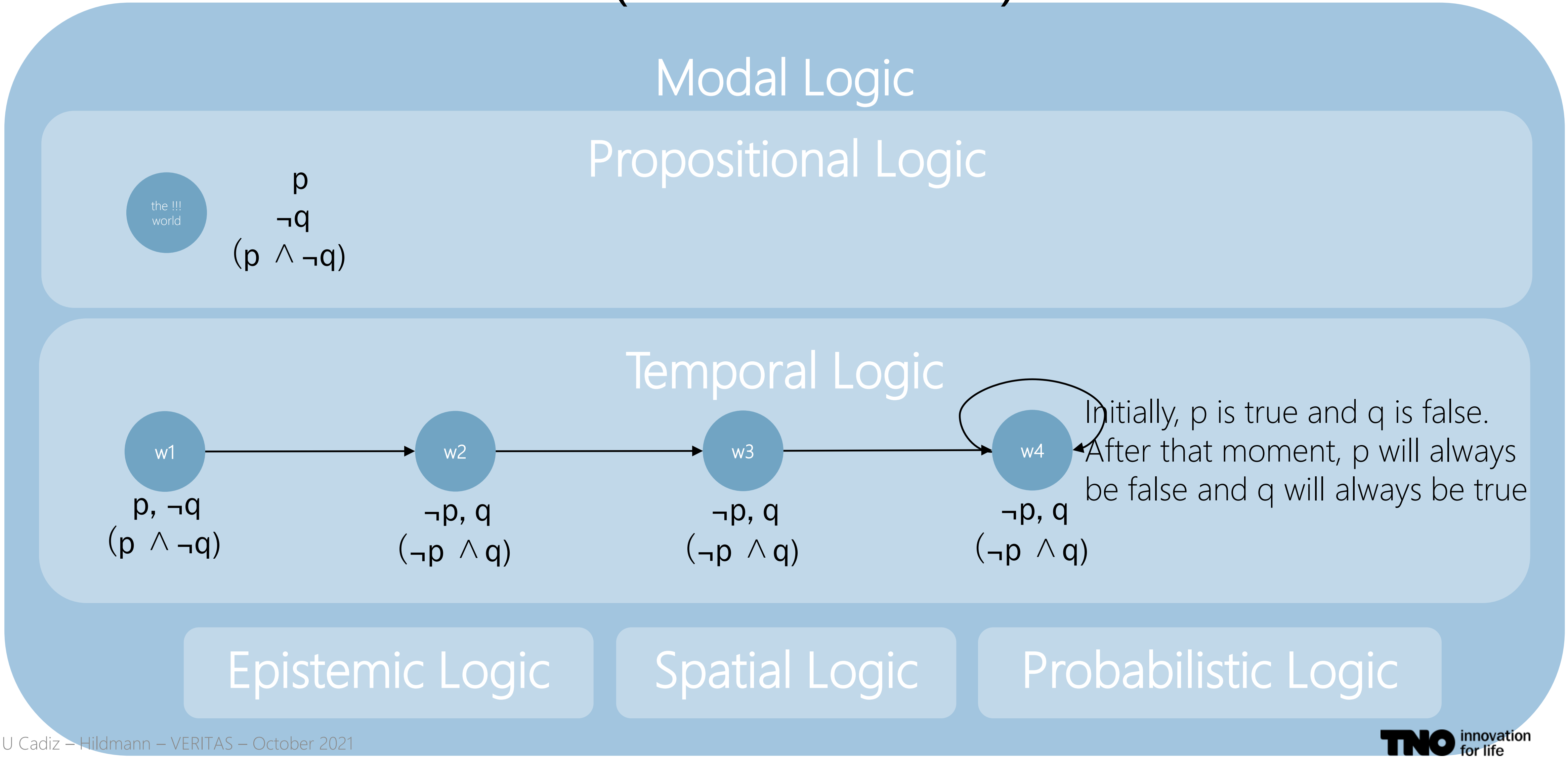
Epistemic Logic

Spatial Logic

Probabilistic Logic

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

the !!! world

$p$

$\neg q$

$(p \land \neg q)$

Temporal Logic

w1

$p, \neg q$

$(p \land \neg q)$

w2

$\neg p, q$

$(\neg p \land q)$

w3

$\neg p, q$

$(\neg p \land q)$

w4

$\neg p, q$

$(\neg p \land q)$

Initially, p is true and q is false. After that moment, p will always be false and q will always be true

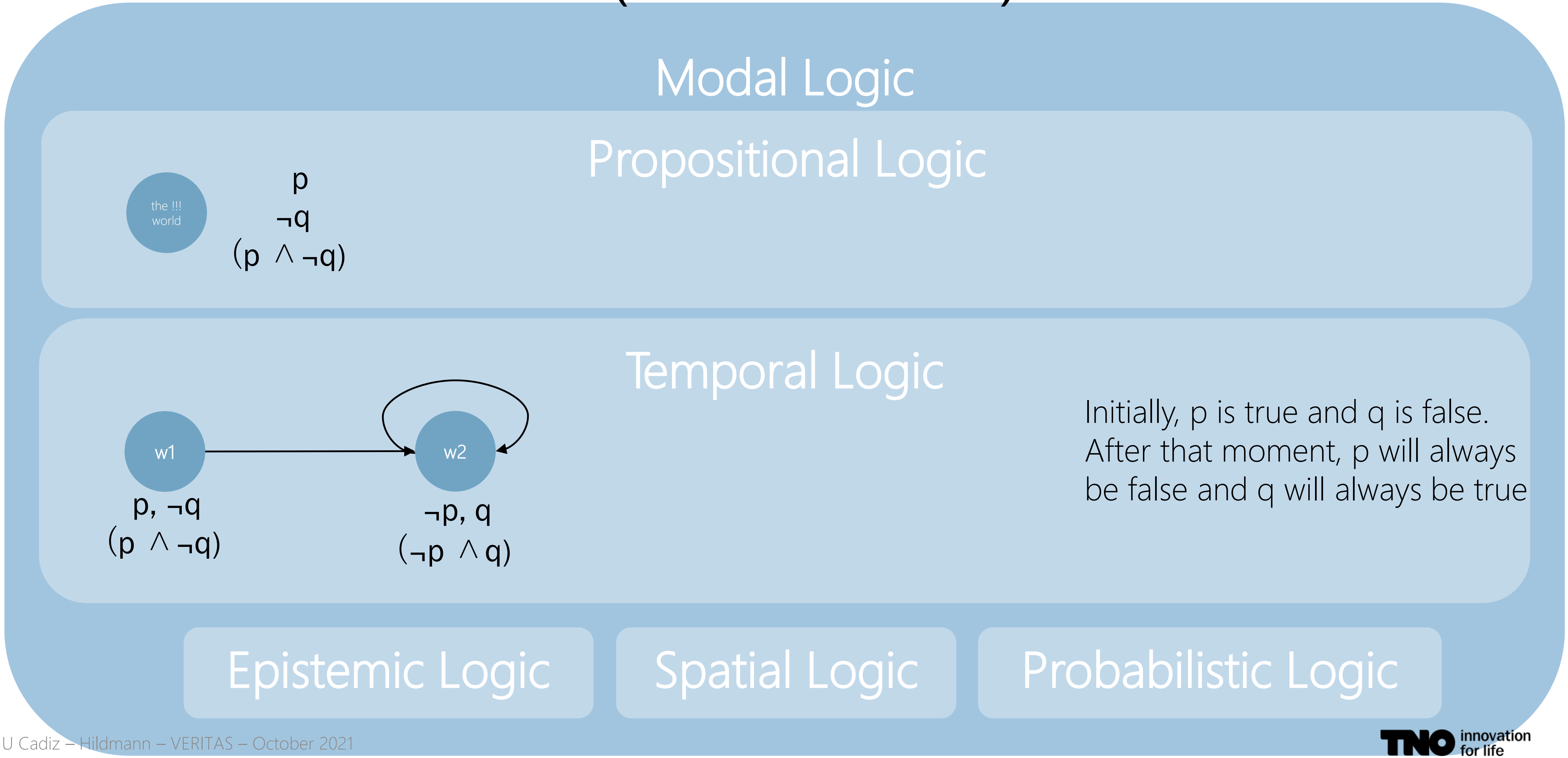
Epistemic Logic

Spatial Logic

Probabilistic Logic

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)



Modal Logic

Propositional Logic

$p$

$\neg q$

$(p \land \neg q)$

Temporal Logic

$p, \neg q$

$(p \land \neg q)$

$\neg p, q$

$(\neg p \land q)$

Initially, p is true and q is false. After that moment, p will always be false and q will always be true

Epistemic Logic

Spatial Logic

Probabilistic Logic

# A TEMPORAL FORMALISM

## LINEAR TEMPORAL LOGIC (A MODAL LOGIC)

Modal Logic

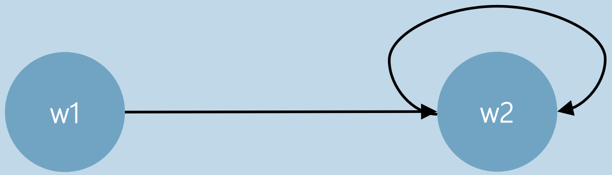
Propositional Logic

$p$

$\neg q$

$(p \land \neg q)$

Temporal Logic



$p, \neg q$

$(p \land \neg q)$

$\neg p, q$

$(\neg p \land q)$

Initially, p is true and q is false. After that moment, p will always be false and q will always be true

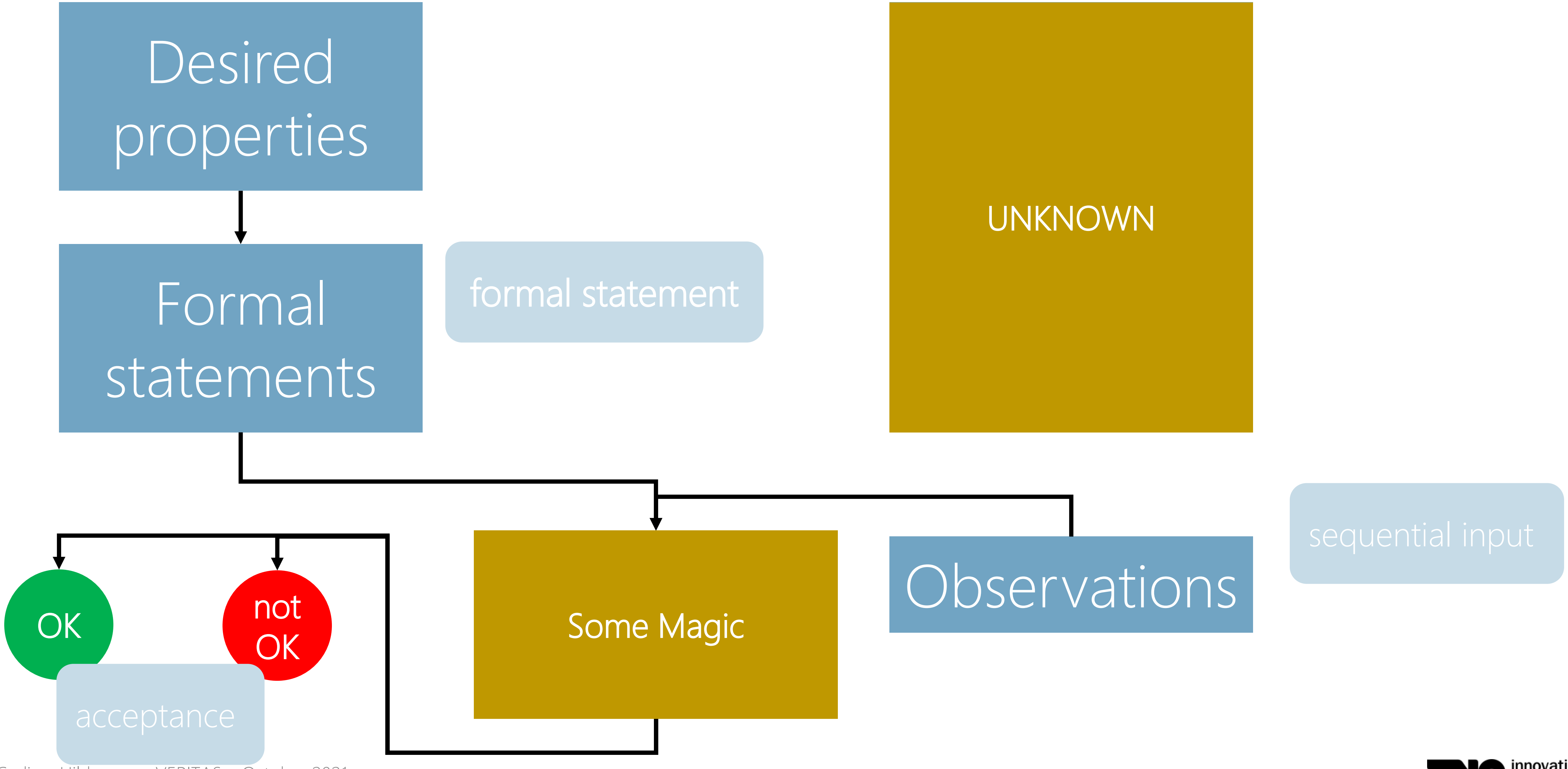
Any statement φ defines an ininite number of models:
The set of models that make the statement true!

Epistemic Logic

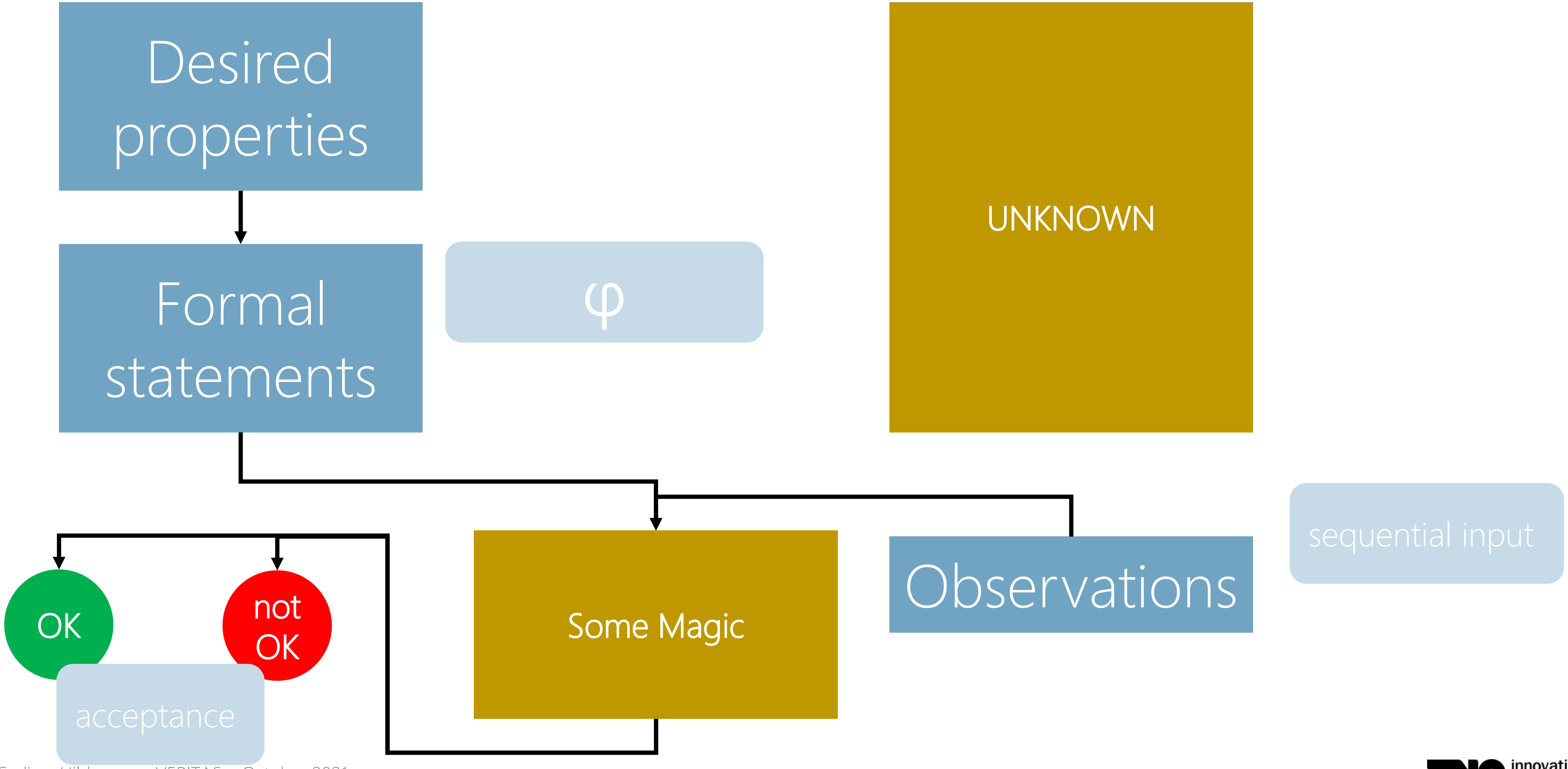
Spatial Logic

Probabilistic Logic

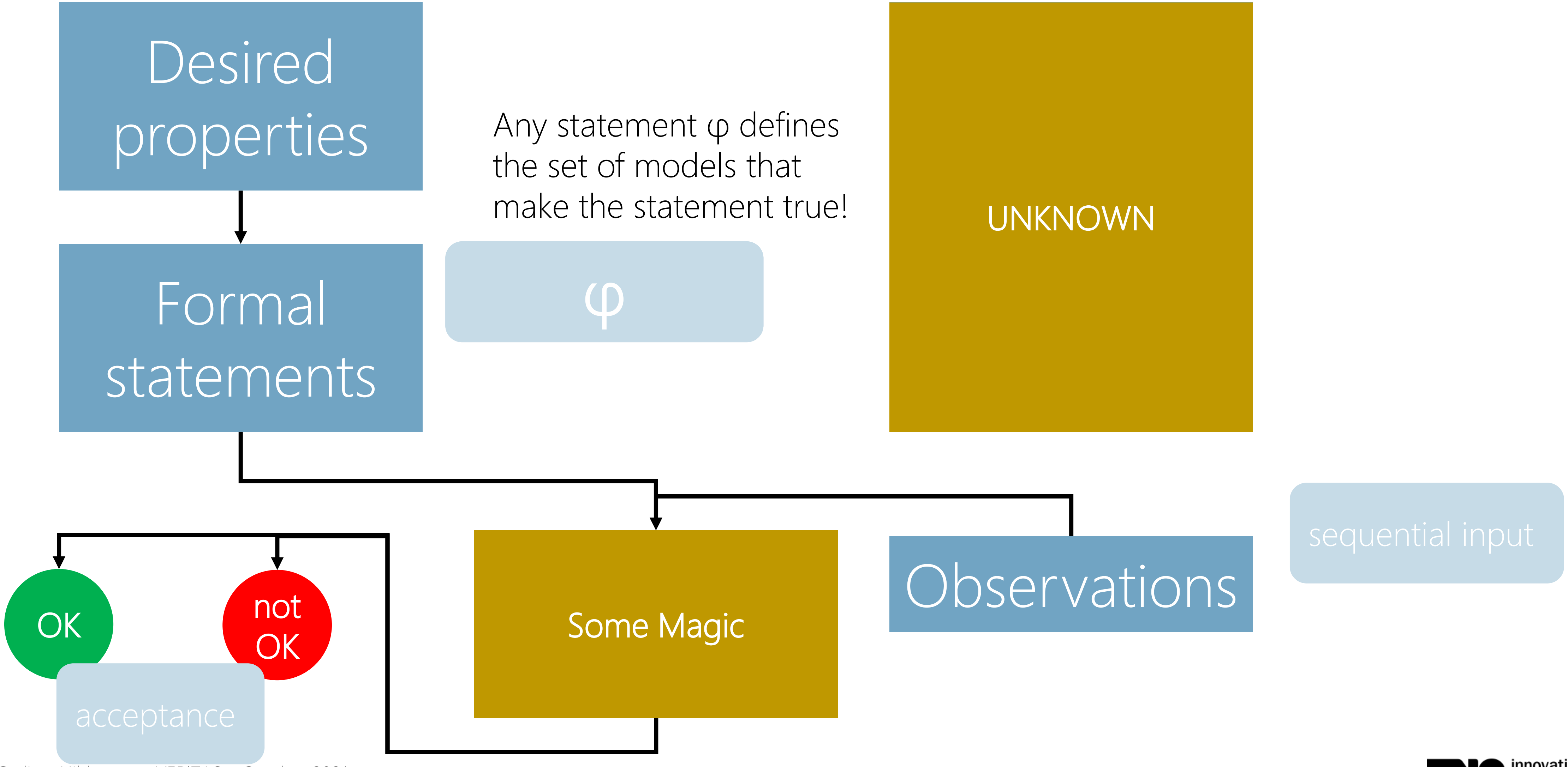
# WHAT WE HAVE


Desired properties
Formal statements
formal statement
UNKNOWN
Some Magic
Observations
sequential input
OK
not OK
acceptance

# WHAT WE HAVE

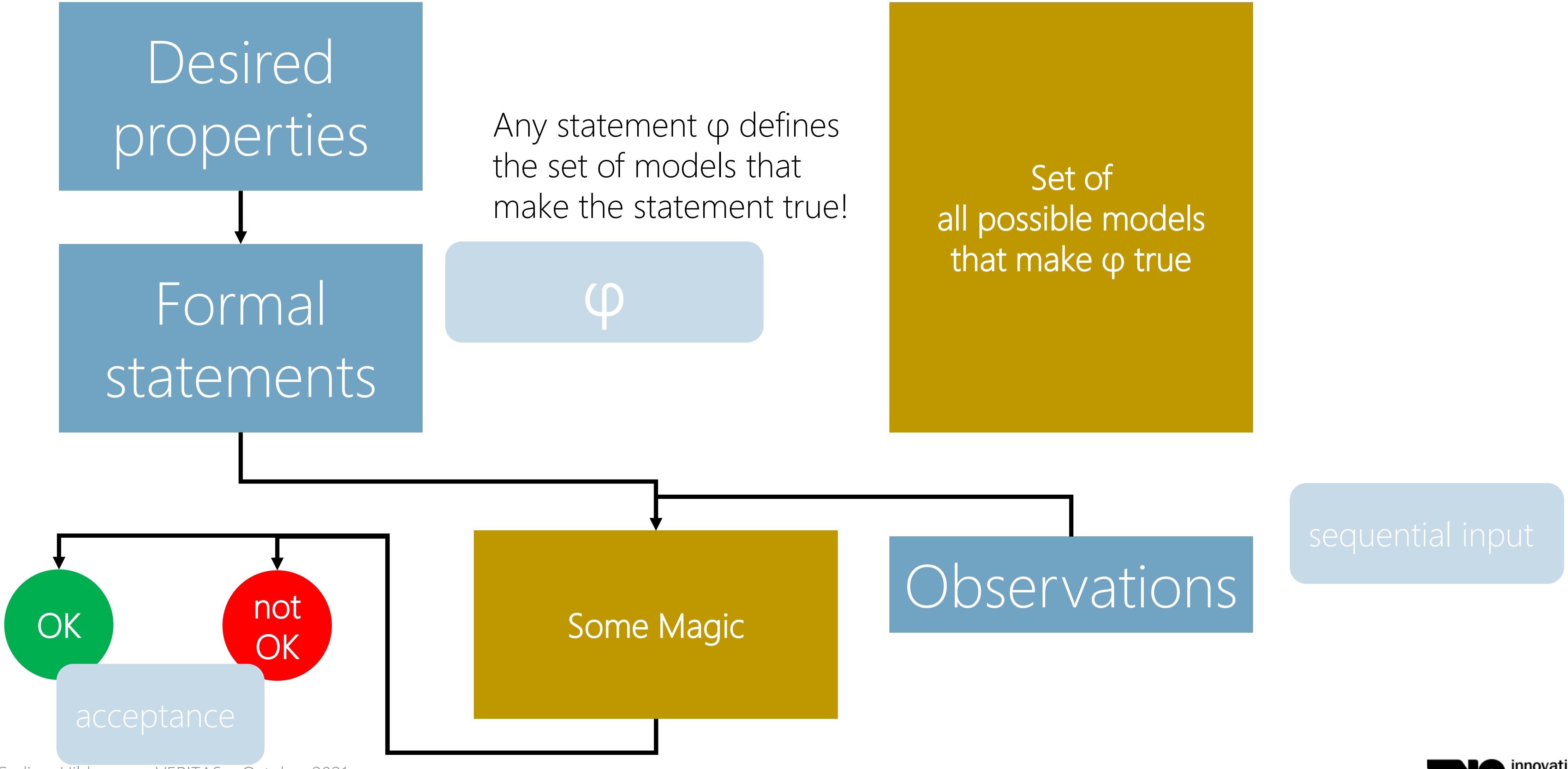

Desired properties
Formal statements
φ
UNKNOWN
Some Magic
Observations
sequential input
OK
not OK
acceptance

# WHAT WE HAVE



Any statement φ defines the set of models that make the statement true!

# WHAT WE HAVE



Desired properties

Formal statements

Any statement φ defines the set of models that make the statement true!

φ

Set of all possible models that make φ true

sequential input

Observations

Some Magic

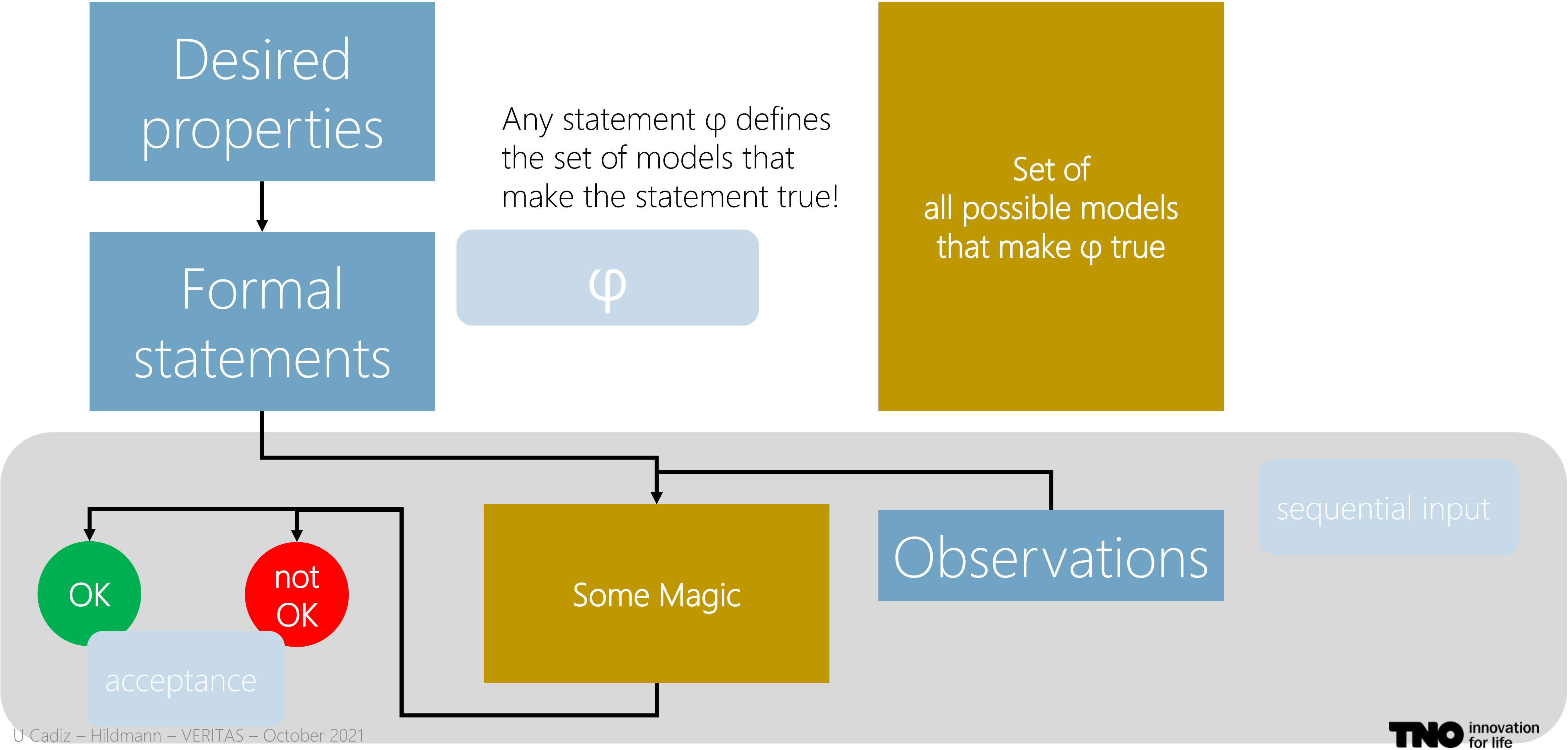
OK

not OK

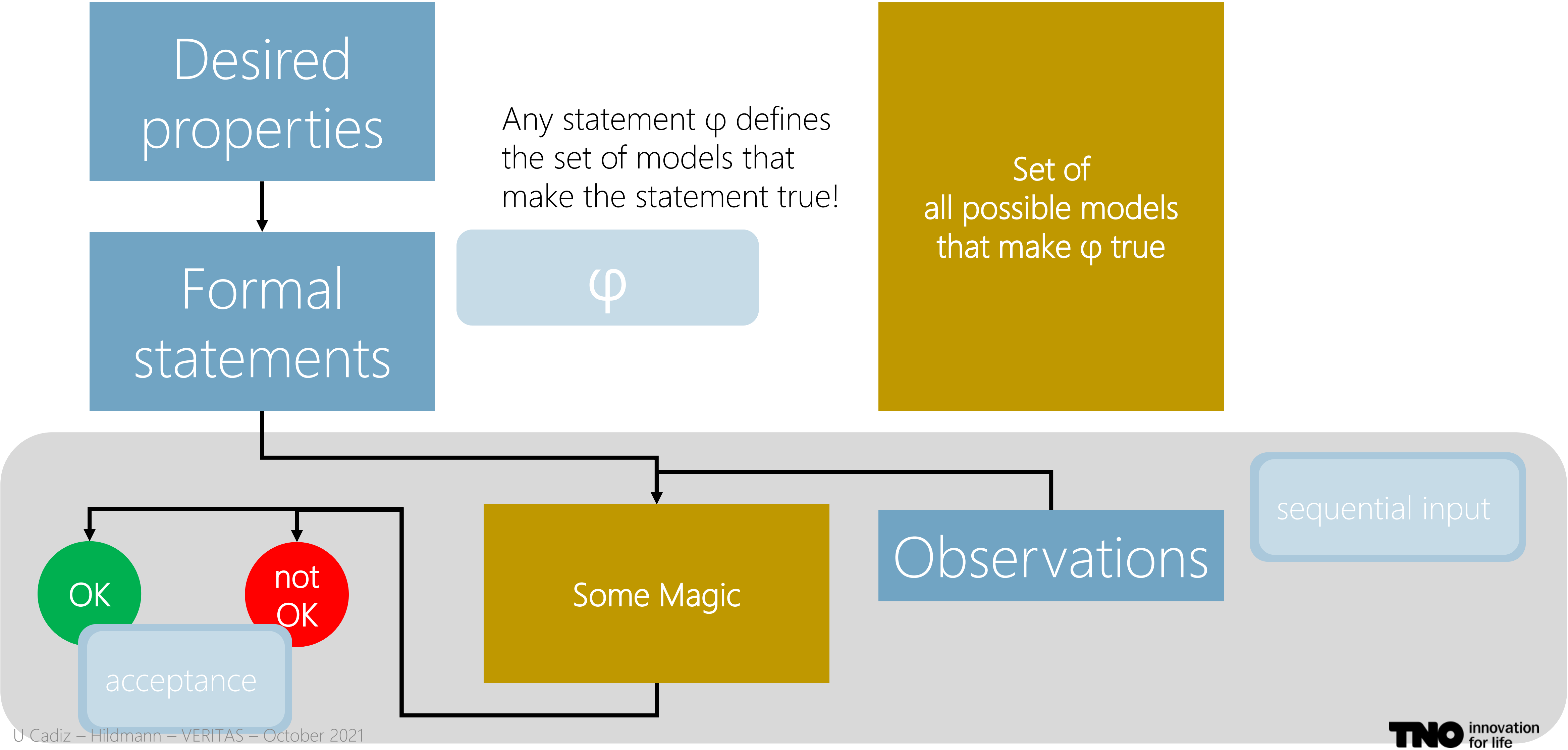
acceptance

# WHAT WE HAVE

Any statement φ defines the set of models that make the statement true!



# WHAT WE HAVE



Desired properties

Formal statements

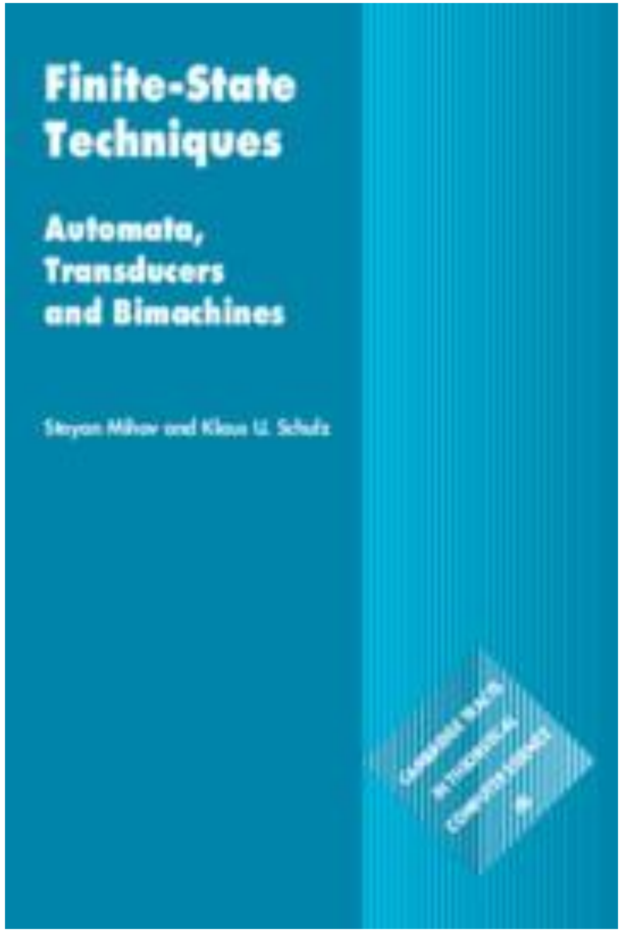
Any statement φ defines the set of models that make the statement true!

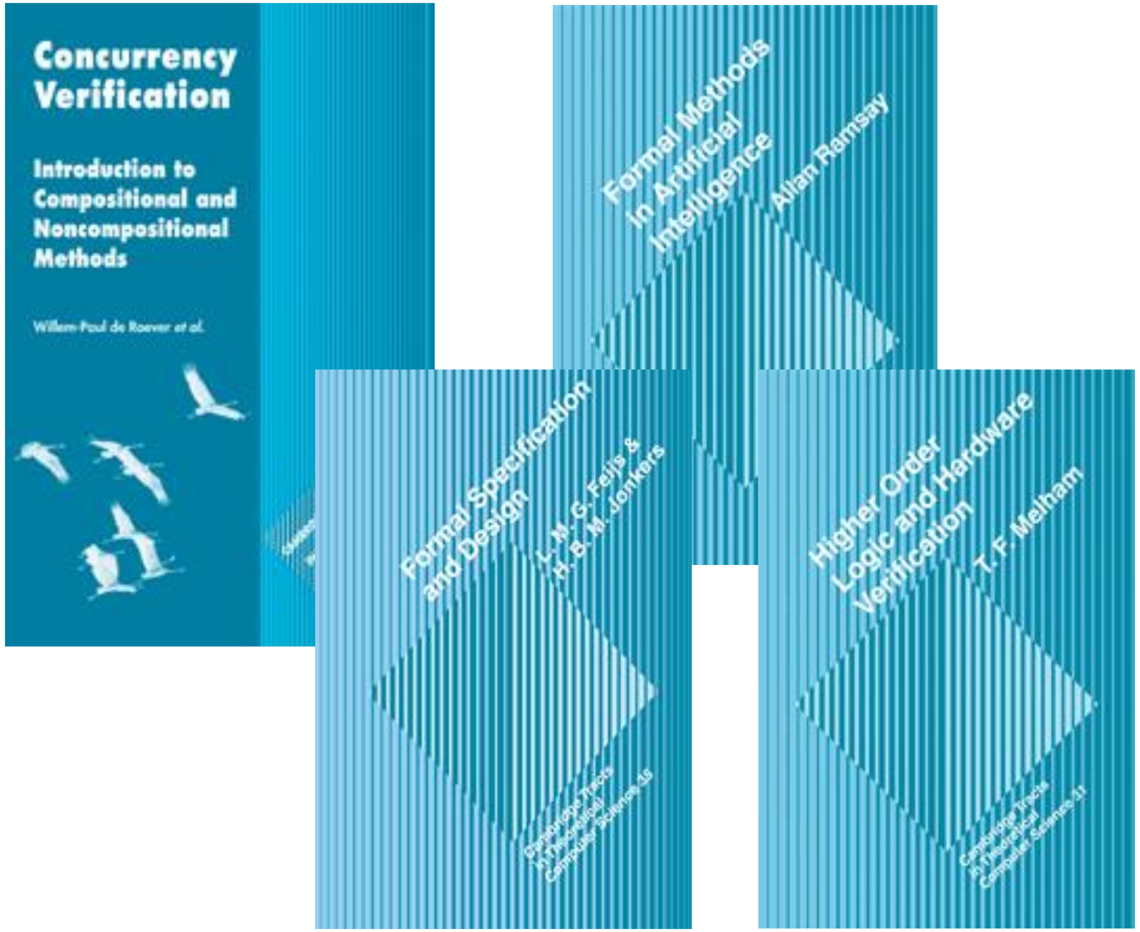
φ

Set of all possible models that make φ true

# › SEQUENTIAL INPUT

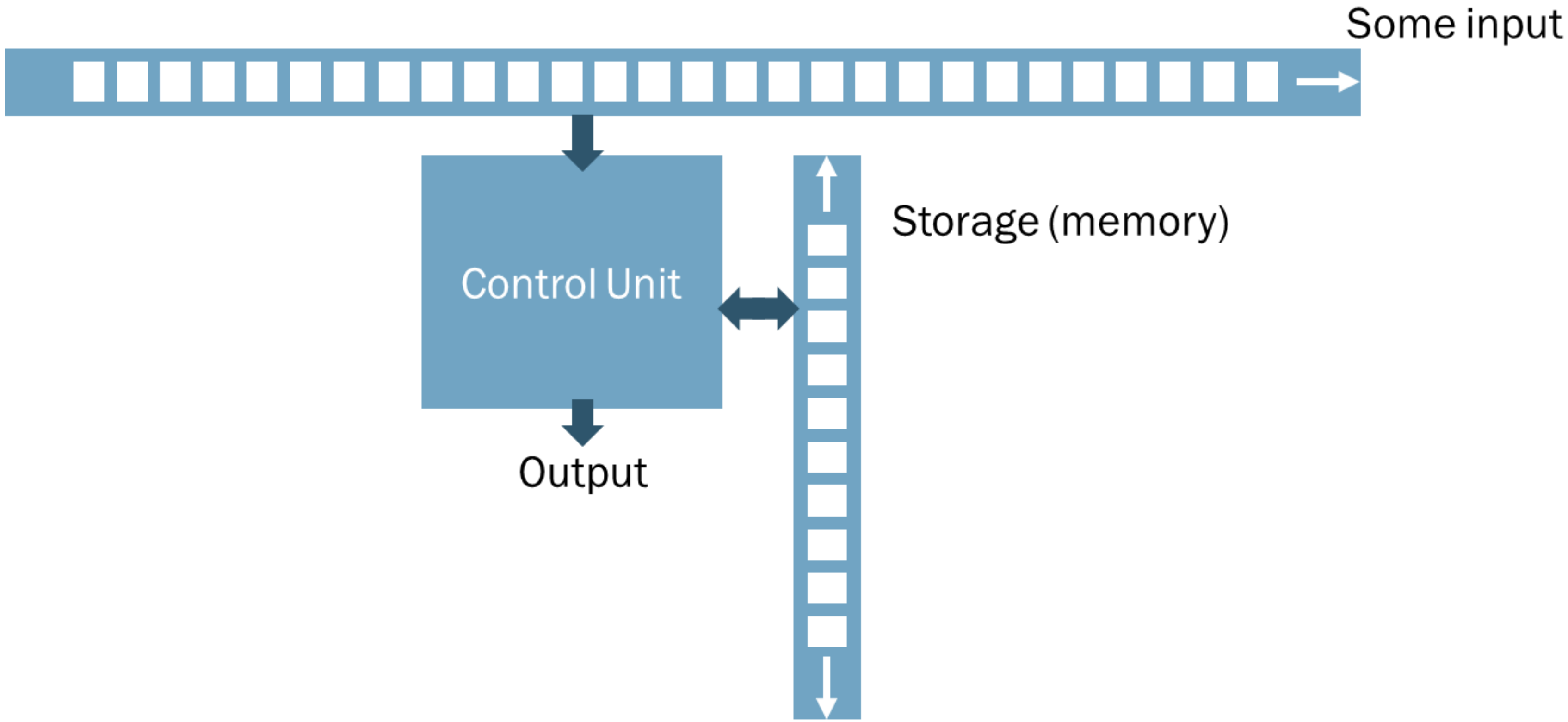
## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA


Finite-State Techniques
Automata, Transducers and Bimachines
Stoyan Mihov and Klaus U. Schulz
Cambridge Tracts in Theoretical Computer Science


Concurrency Verification
Introduction to Compositional and Noncompositional Methods
Willem-Paul de Roever et al.
Formal Methods in Artificial Intelligence
Allan Ramsay
Formal Specification and Design
L. M. G. Feijs & H. B. M. Jonkers
Cambridge Tracts in Theoretical Computer Science 35
Higher Order Logic and Hardware Verification
T. F. Melham
Cambridge Tracts in Theoretical Computer Science 31

# SEQUENTIAL INPUT

## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA



# SEQUENTIAL INPUT

## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

Regular expressions

# SEQUENTIAL INPUT

## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

### Regular expressions

(a)* bbb (c)*

# SEQUENTIAL INPUT

## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

### Regular expressions

(a)* bbb (c)* examples: bbb, abbbc, aaaaaaabbbc, aabbbccccccccc, …

# SEQUENTIAL INPUT

## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

### Regular expressions

| | |
|---|---|
| (a)* bbb (c)* | examples: bbb, abbbc, aaaaaaabbbc, aabbbccccccccc, … |
| (a+b) b (a+c) | examples: aba, abc, bba, bbc |

# SEQUENTIAL INPUT

## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

### Regular expressions

| | |
|---|---|
| (a)* bbb (c)* | examples: bbb, abbbc, aaaaaaabbbc, aabbbccccccccc, … |
| (a+b) b (a+c) | examples: aba, abc, bba, bbc |
| (a(a)* b(b)*)* | examples: ab, aab, aaab, aabaab, aaabbaab, aaaaaabbbb, … |

# SEQUENTIAL INPUT

## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

### Regular expressions

| | |
|---|---|
| (a)* bbb (c)* | examples: bbb, abbbc, aaaaaaabbbc, aabbbccccccccc, … |
| (a+b) b (a+c) | examples: aba, abc, bba, bbc |
| (a(a)* b(b)*)* | examples: ab, aab, aaab, aabaab, aaabbaab, aaaaaabbbb, … |

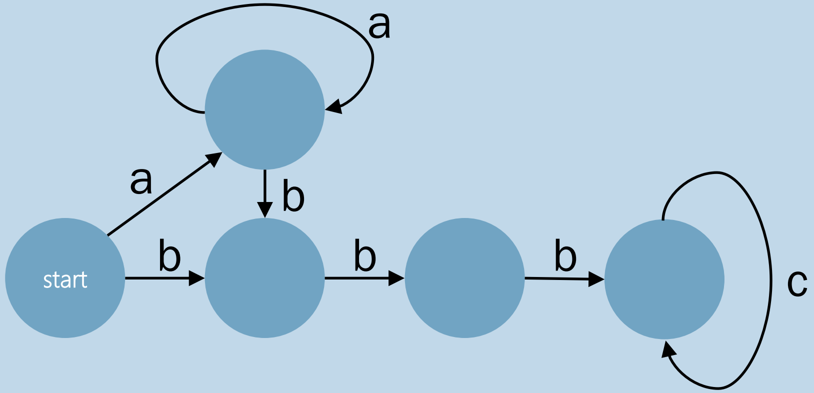
### Automata

# SEQUENTIAL INPUT
## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

### Regular expressions

| | |
|---|---|
| (a)* bbb (c)* | examples: bbb, abbbc, aaaaaaabbbc, aabbbccccccc, ... |
| (a+b) b (a+c) | examples: aba, abc, bba, bbc |
| (a(a)* b(b)*)* | examples: ab, aab, aaab, aabaab, aaabbaab, aaaaaabbbb, ... |

### Automata



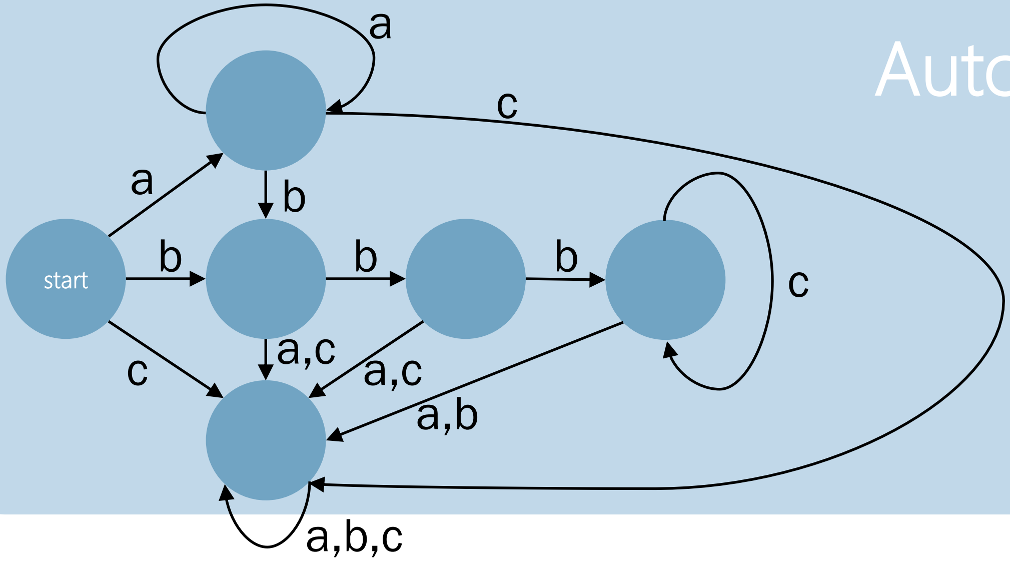
# SEQUENTIAL INPUT

## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

### Regular expressions

| | |
|---|---|
| ➡ (a)* bbb (c)* | examples: bbb, abbbc, aaaaaaabbbc, aabbbccccccccc, … |
| (a+b) b (a+c) | examples: aba, abc, bba, bbc |
| (a(a)* b(b)*)* | examples: ab, aab, aaab, aabaab, aaabbaab, aaaaaabbbb, … |

### Automata



# SEQUENTIAL INPUT
## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

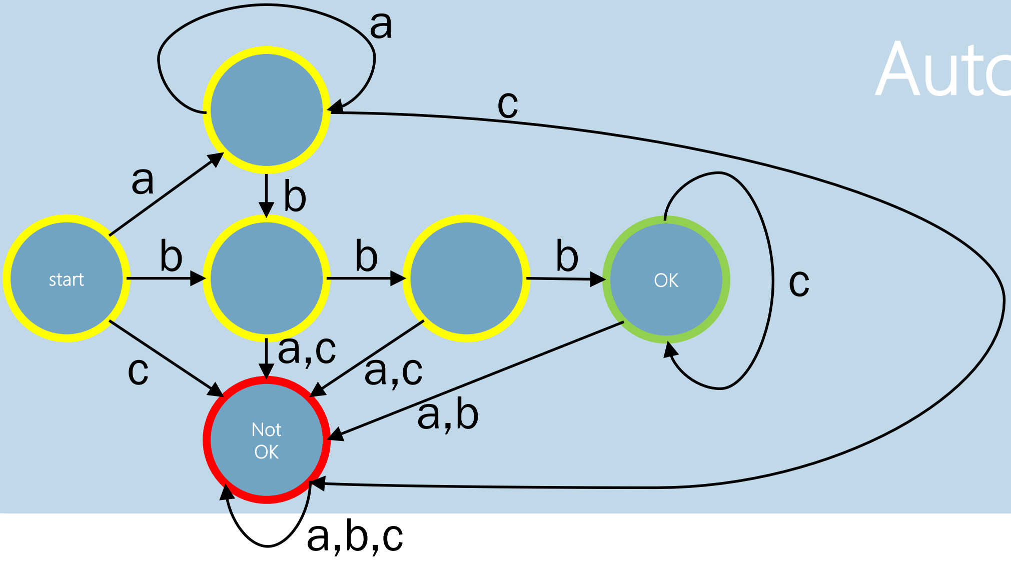
### Regular expressions

➡ (a)* bbb (c)* examples: bbb, abbbc, aaaaaaabbbc, aabbbccccccccc, …

(a+b) b (a+c) examples: aba, abc, bba, bbc

(a(a)* b(b)*)* examples: ab, aab, aaab, aabaab, aaabbaab, aaaaaabbbb, …

### Automata



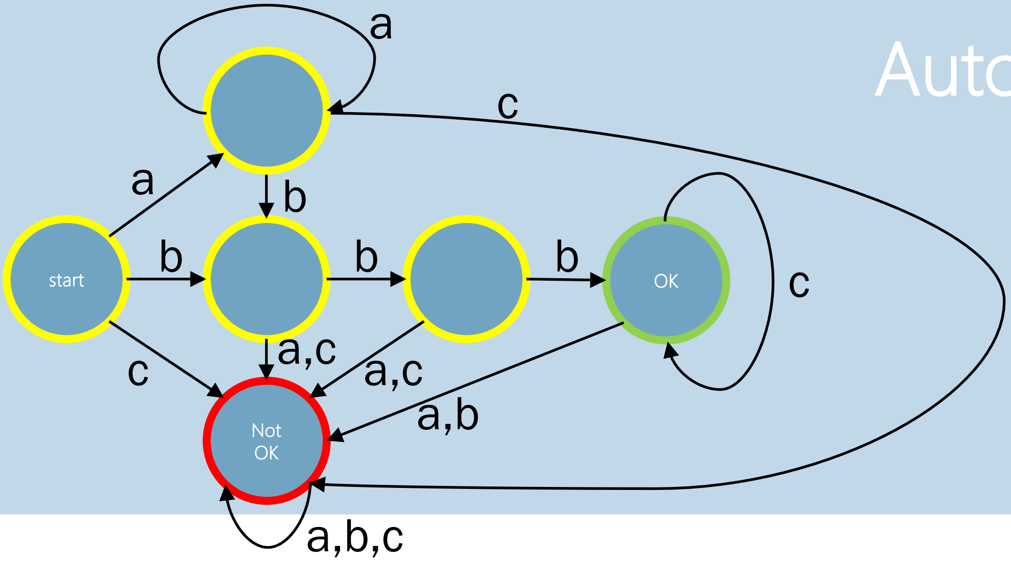
# SEQUENTIAL INPUT

## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

### Regular expressions

| | |
|---|---|
| ➡ (a)* bbb (c)* | examples: bbb, abbbc, aaaaaaabbbc, aabbbccccccccc, … |
| (a+b) b (a+c) | examples: aba, abc, bba, bbc |
| (a(a)* b(b)*)* | examples: ab, aab, aaab, aabaab, aaabbaab, aaaaaabbbb, … |

### Automata



A regular expression defines
the set of automata that accept it
and
Any automata defines a language
that can be expressed by a RegEx

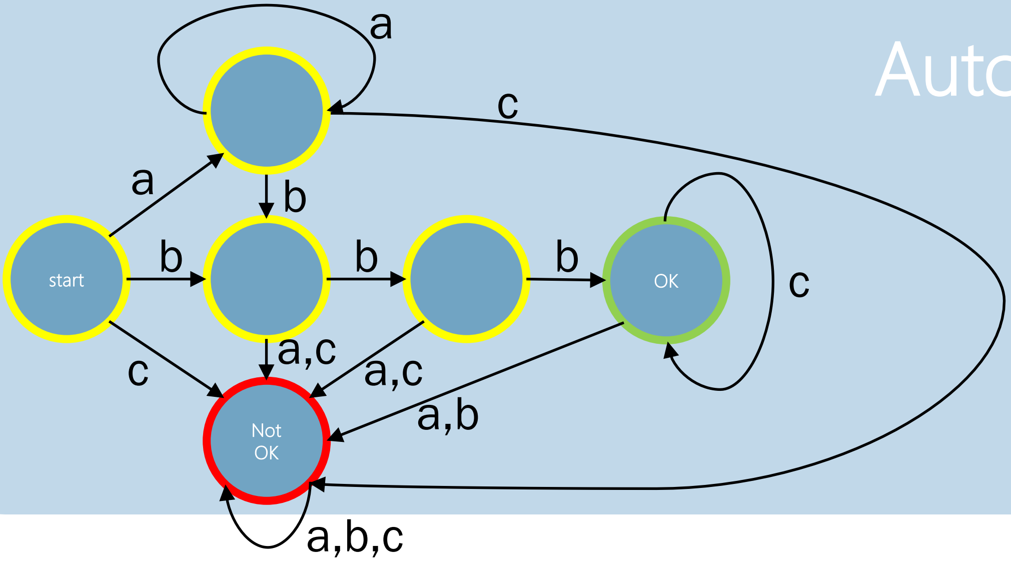
# SEQUENTIAL INPUT

## REGULAR EXPRESSIONS AND ACCEPTING AUTOMATA

### Regular expressions

| | |
|---|---|
| (a)* bbb (c)* | examples: bbb, abbbc, aaaaaaabbbc, aabbbccccccccc, … |
| (a+b) b (a+c) | examples: aba, abc, bba, bbc |
| (a(a)* b(b)*)* | examples: ab, aab, aaab, aabaab, aaabbaab, aaaaaabbbb, … |

### Automata



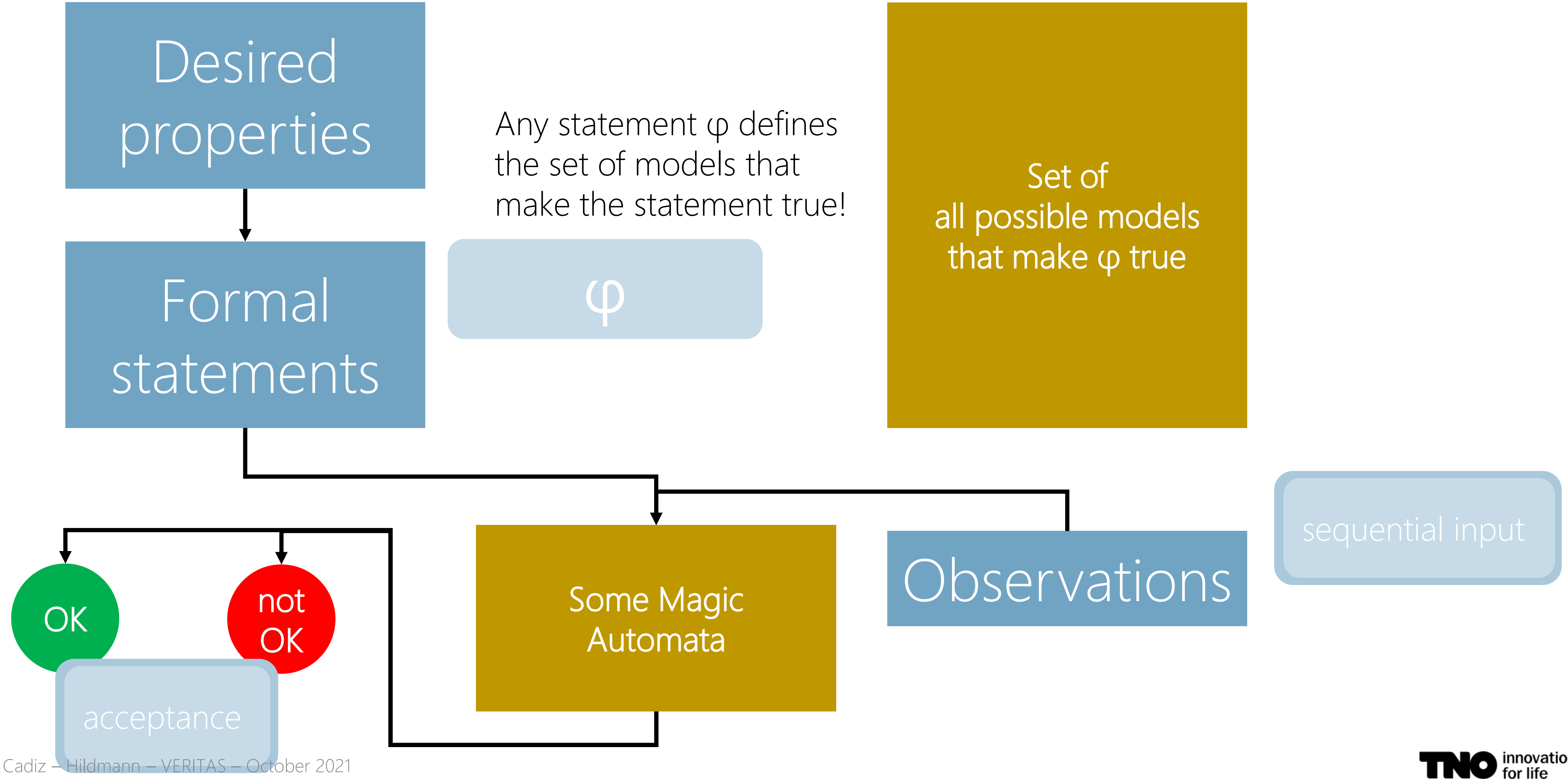
A regular expression defines the set of automata that accept it

and

Any automata defines a language that can be expressed by a RegEx

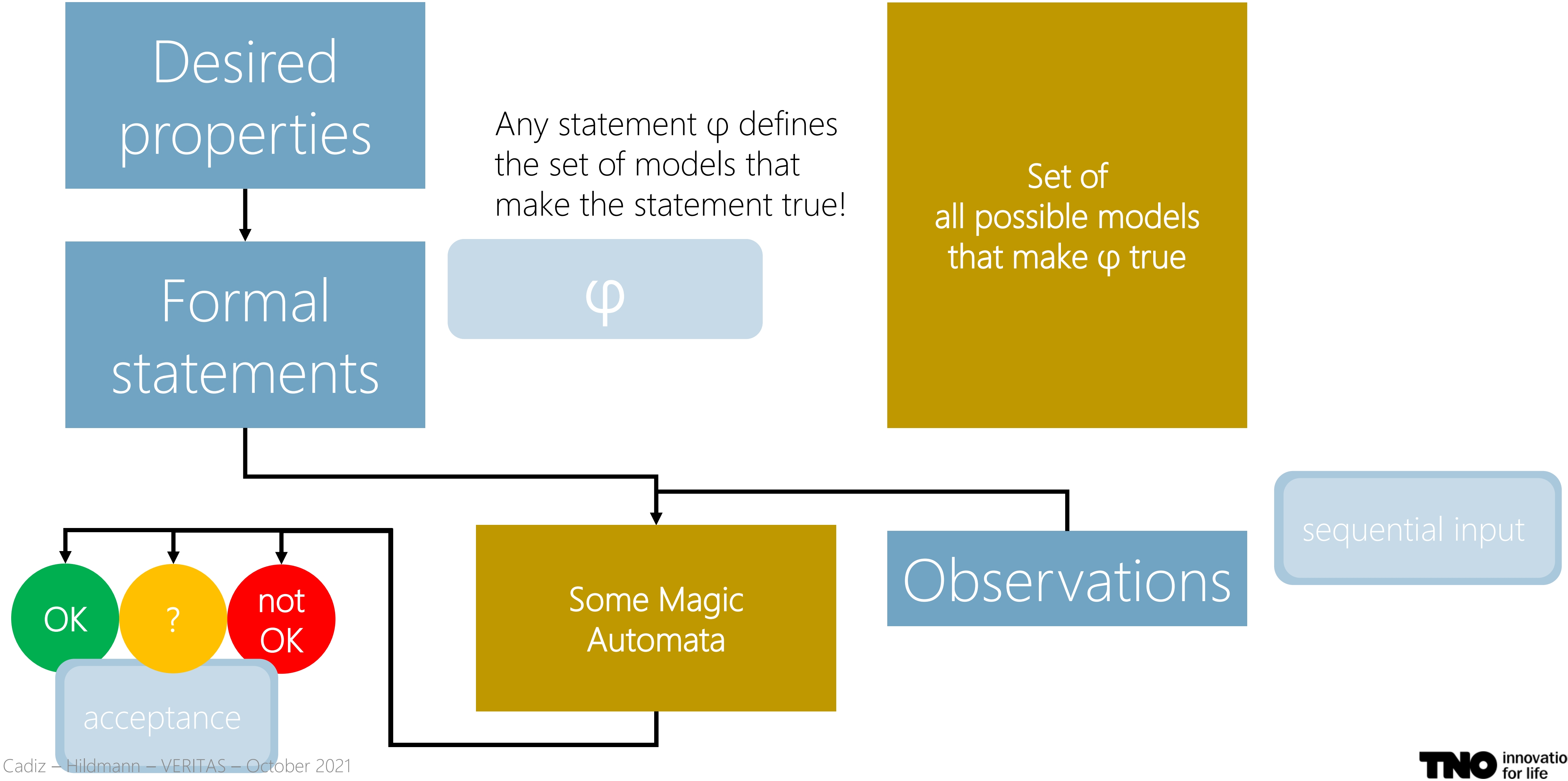
A quick word about infinite things …

# WHAT WE CAN DO



Desired properties
Formal statements
Any statement φ defines the set of models that make the statement true!
φ
Set of all possible models that make φ true
Some Magic Automata
Observations
sequential input
OK
not OK
acceptance

# WHAT WE CAN DO



Desired properties

Formal statements

Any statement φ defines the set of models that make the statement true!

φ

Set of all possible models that make φ true

Some Magic Automata

Observations

sequential input

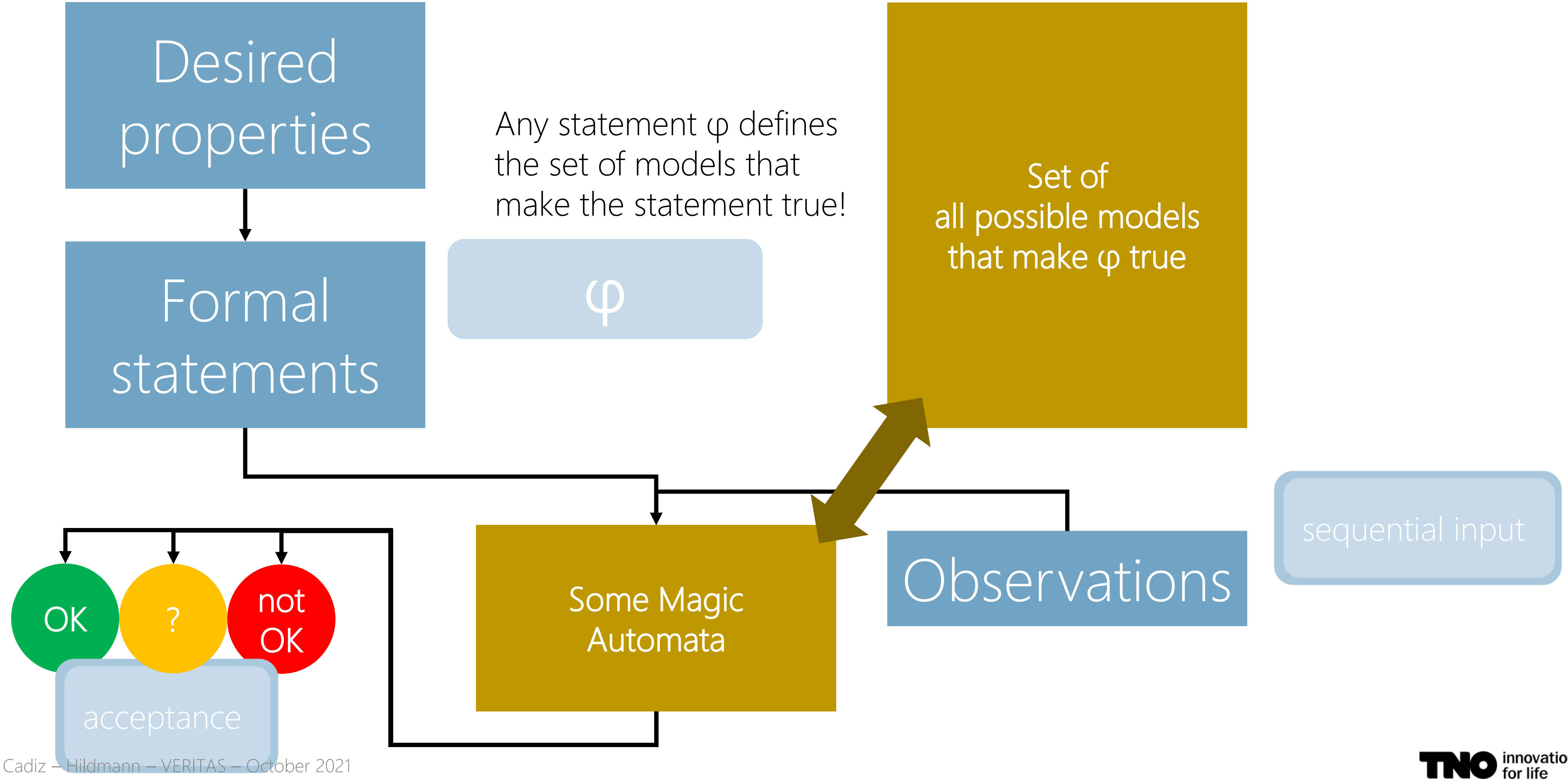
OK

?

not OK

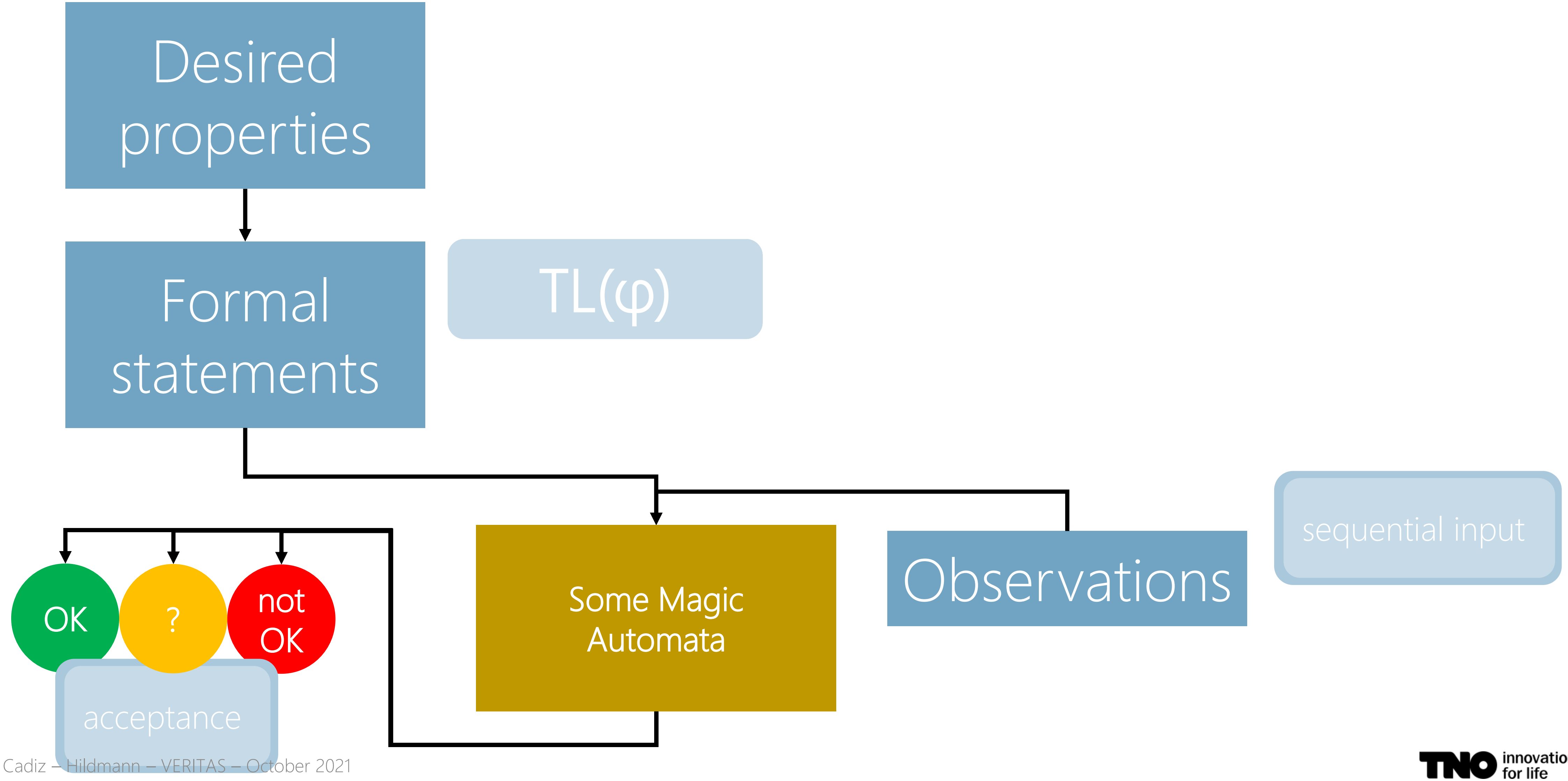
acceptance

# WHAT WE CAN DO



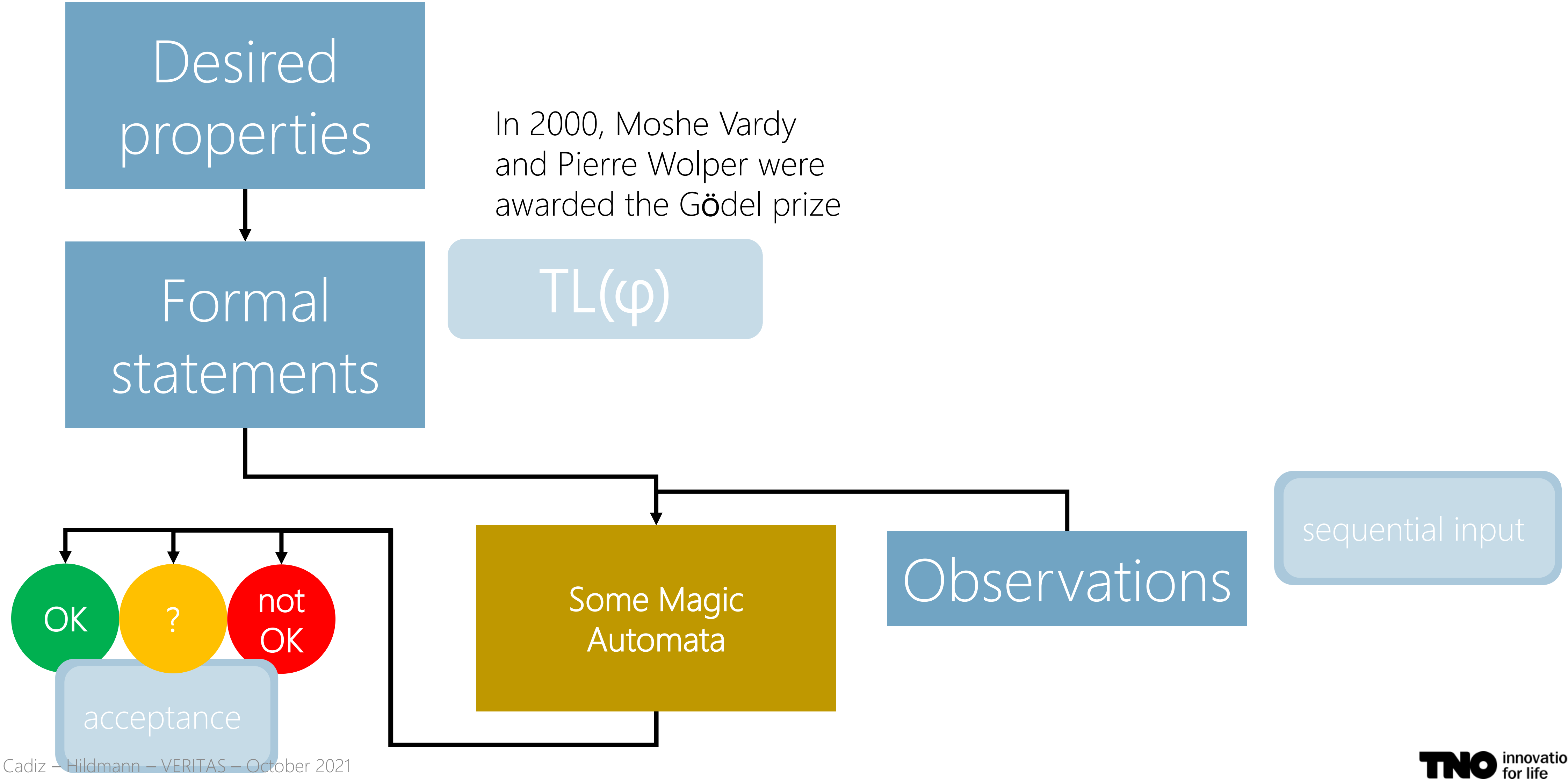
Any statement φ defines the set of models that make the statement true!

# WHAT WE CAN DO

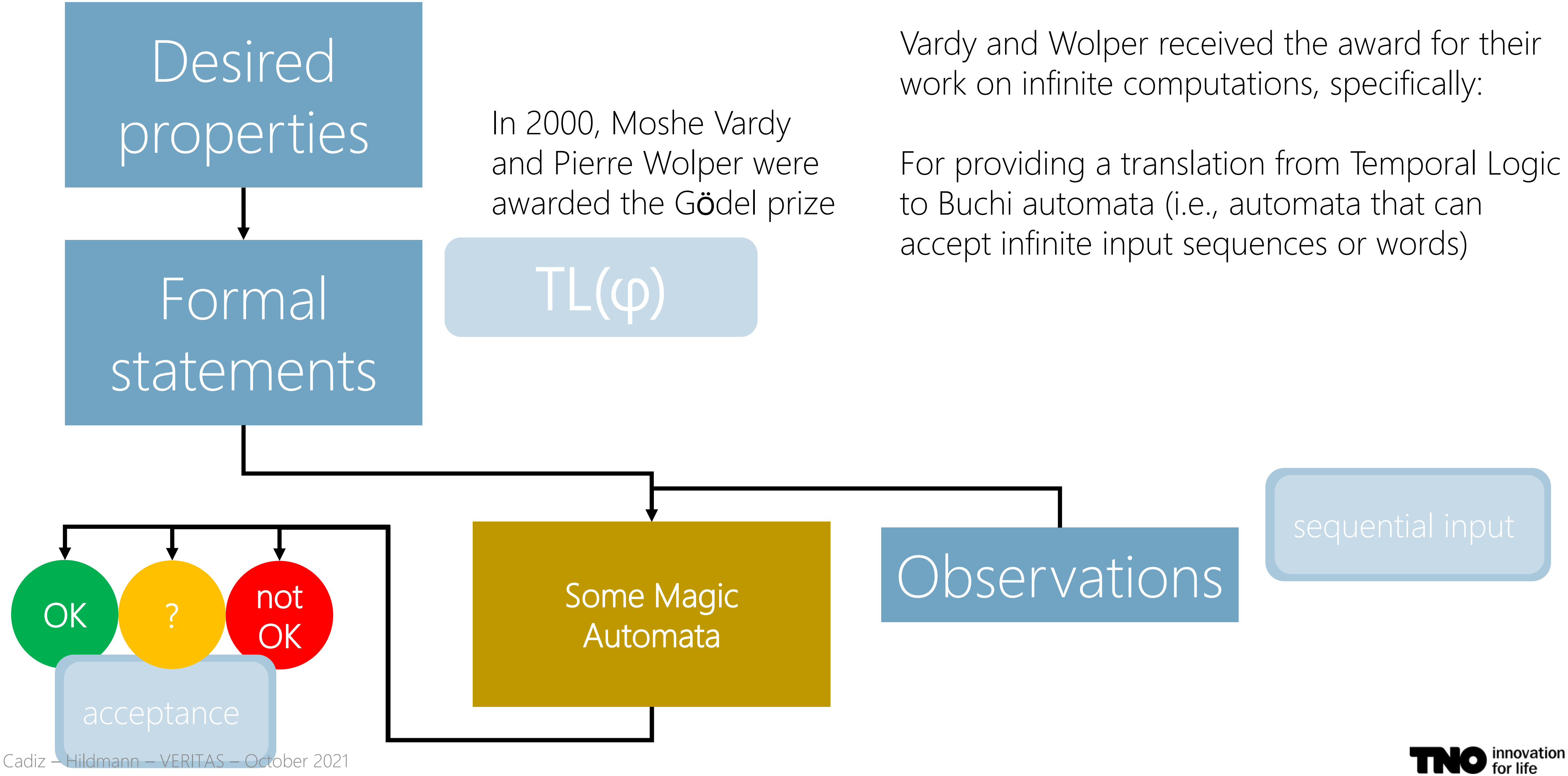


# WHAT WE CAN DO

In 2000, Moshe Vardy and Pierre Wolper were awarded the Gödel prize



# WHAT WE CAN DO

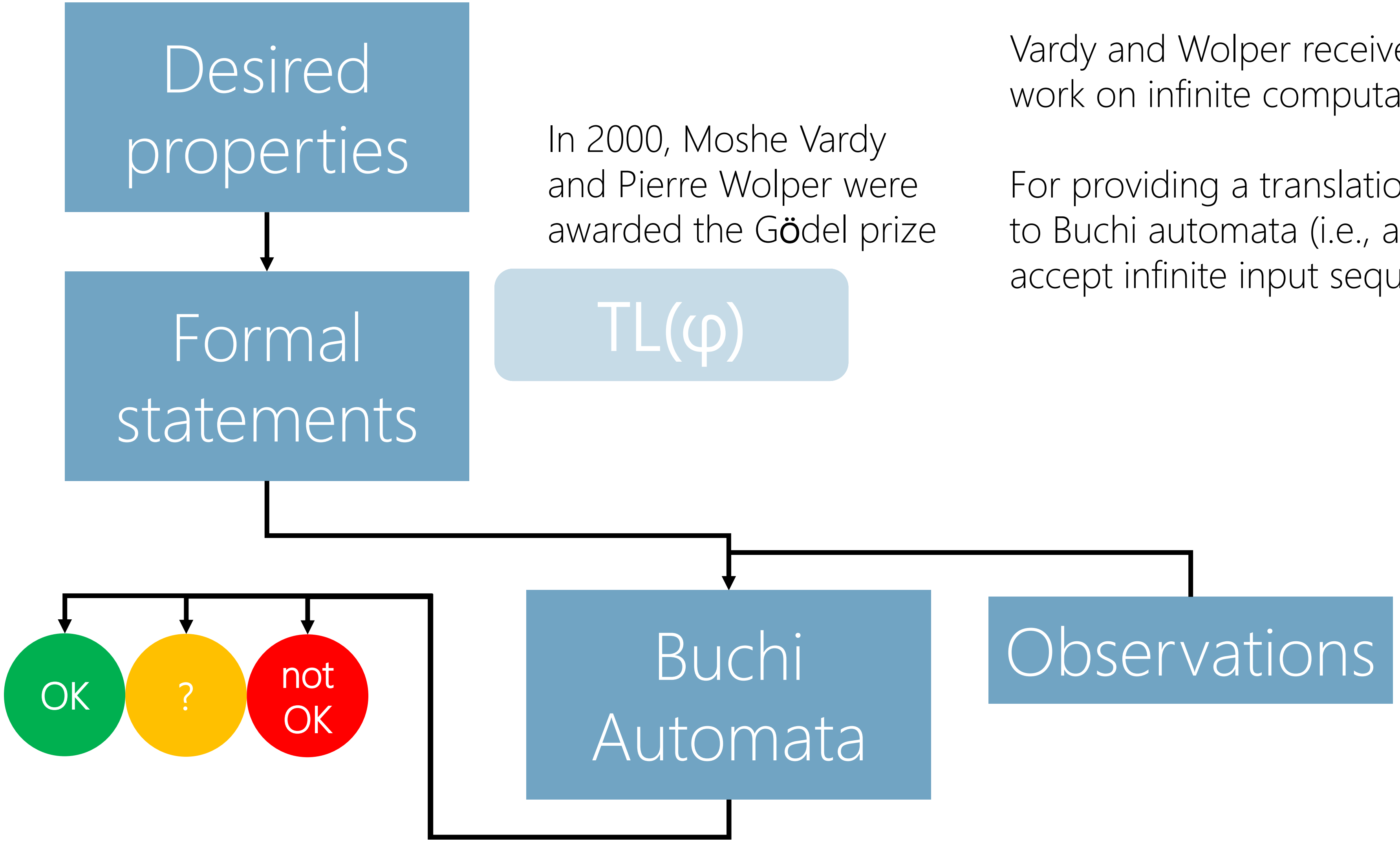


In 2000, Moshe Vardy and Pierre Wolper were awarded the Gödel prize

Vardy and Wolper received the award for their work on infinite computations, specifically:

For providing a translation from Temporal Logic to Buchi automata (i.e., automata that can accept infinite input sequences or words)

# WHAT WE CAN DO



In 2000, Moshe Vardy and Pierre Wolper were awarded the Gödel prize

Vardy and Wolper received the award for their work on infinite computations, specifically:

For providing a translation from Temporal Logic to Buchi automata (i.e., automata that can accept infinite input sequences or words)

# VERITAS

ve•ri•tas

[wey-ri-tahs; *Eng.* ver-i-tas, -tahs]

*-noun Latin.*

TRUTH.

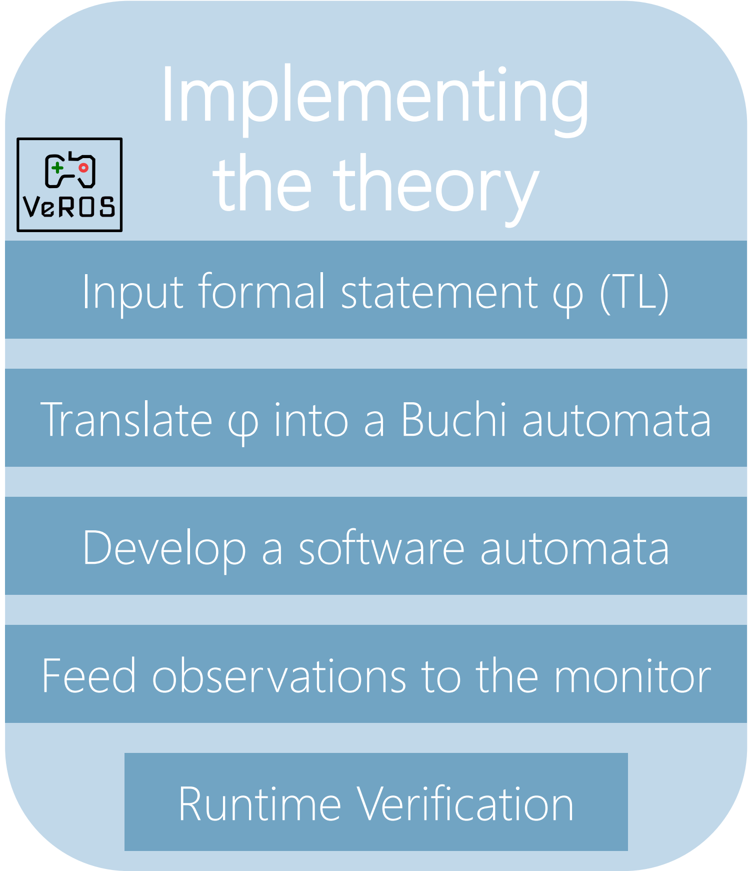
# VERITAS

ve·ri·tas

[wey-ri-tahs; *Eng.* ver-i-tas, -tahs]

*-noun Latin.*

TRUTH.



Implementing the theory

Input formal statement φ (TL)

Translate φ into a Buchi automata

Develop a software automata

Feed observations to the monitor

Runtime Verification

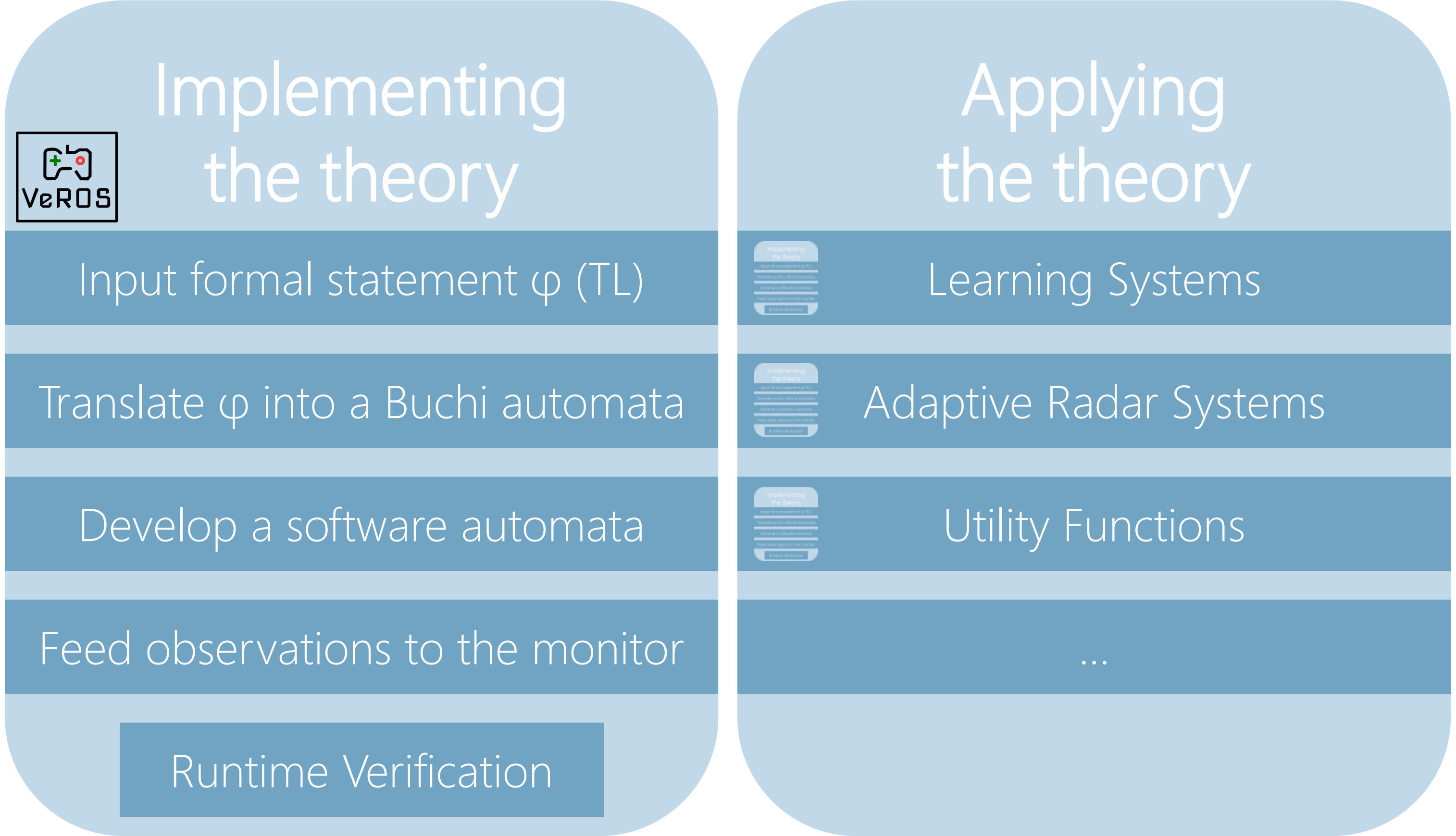
# VERITAS

ve•ri•tas

[wey-ri-tahs; *Eng.* ver-i-tas, -tahs]

*-noun Latin.*

TRUTH.



Implementing the theory

VeROS

Input formal statement φ (TL)

Translate φ into a Buchi automata

Develop a software automata

Feed observations to the monitor

Runtime Verification

Applying the theory

Learning Systems

Adaptive Radar Systems

Utility Functions

…

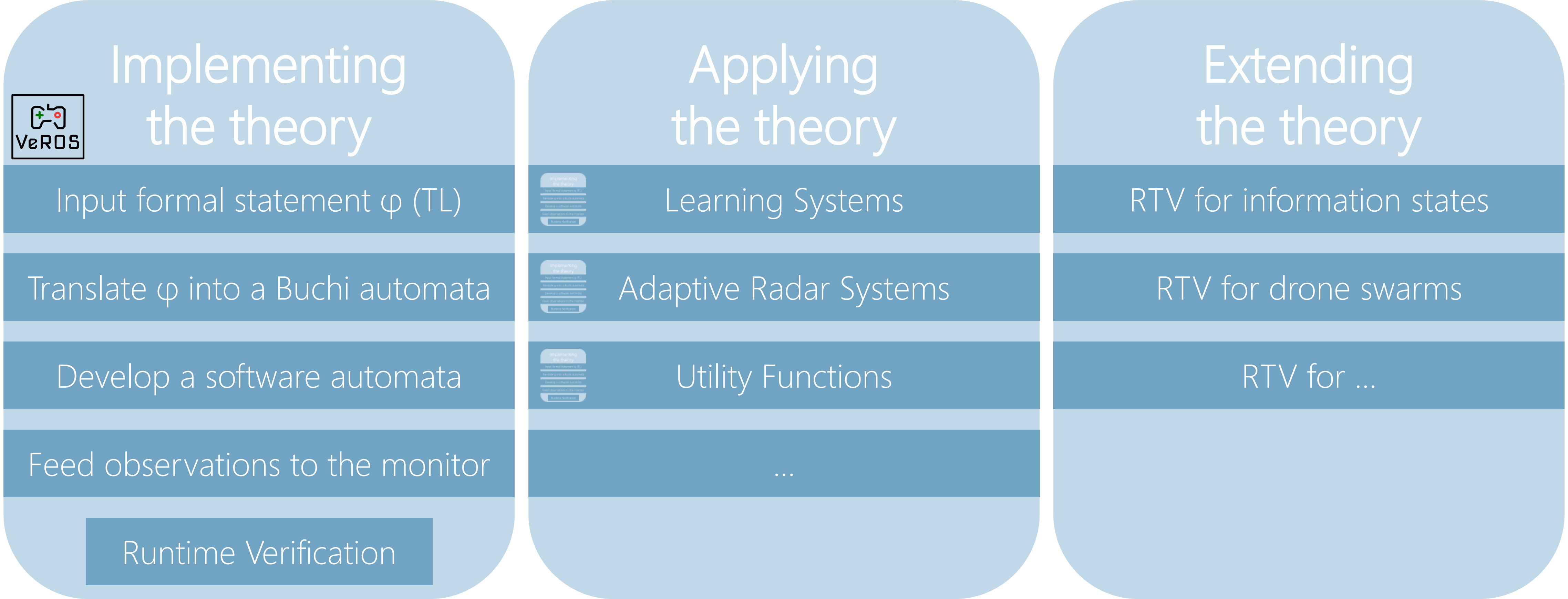
# VERITAS

ve·ri·tas

[wey-ri-tahs; *Eng.* ver-i-tas, -tahs]

*-noun Latin.*

TRUTH.



## Implementing the theory

VeROS

- Input formal statement φ (TL)
- Translate φ into a Buchi automata
- Develop a software automata
- Feed observations to the monitor
- Runtime Verification

## Applying the theory

- Learning Systems
- Adaptive Radar Systems
- Utility Functions
- …

## Extending the theory

- RTV for information states
- RTV for drone swarms
- RTV for …

# THANK YOU FOR YOUR ATTENTION

ANY QUESTIONS?

General, you are listening to a machine!
MOVIECLIPS