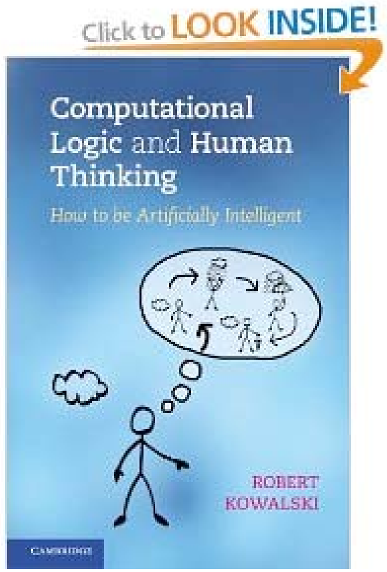
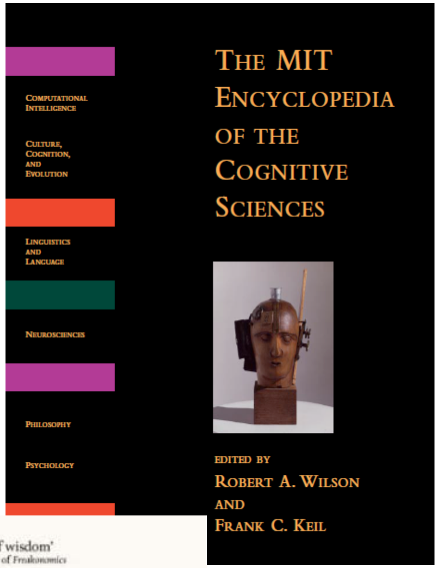
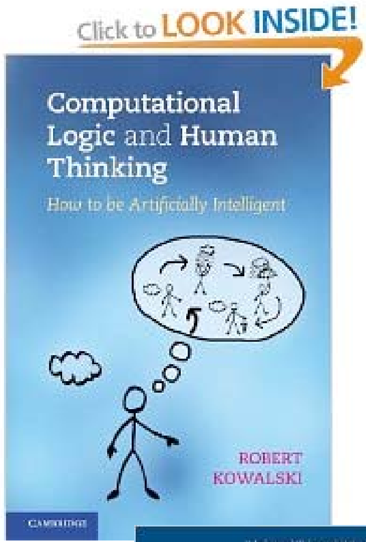
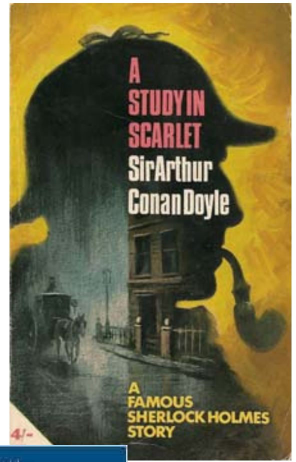
# Computational Logic, Human Thinking and Action

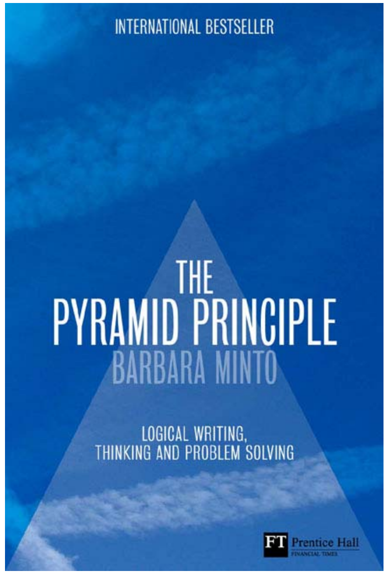
Robert Kowalski
Imperial College London

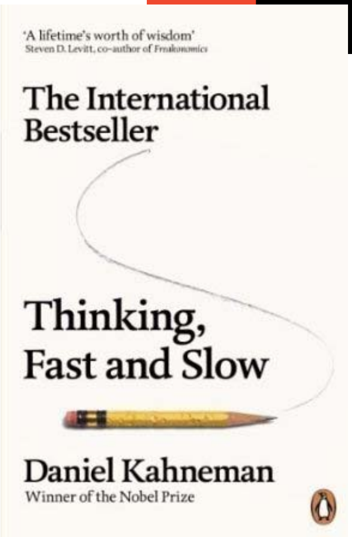


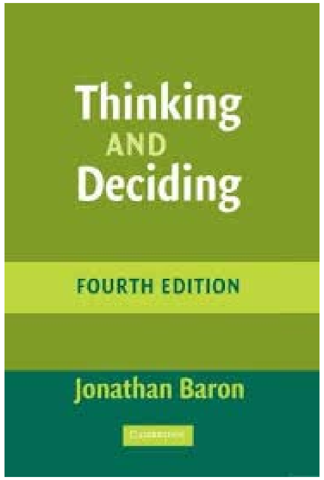

The MIT Encyclopedia of the Cognitive Sciences
Computational Intelligence
Culture, Cognition, and Evolution
Linguistics and Language
Neurosciences
Philosophy
Psychology
Edited by
Robert A. Wilson
and
Frank C. Keil

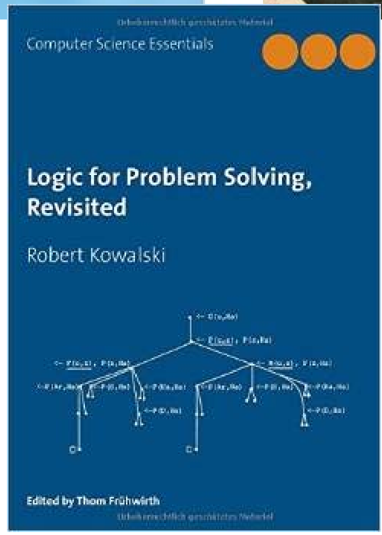

Click to LOOK INSIDE!
Computational Logic and Human Thinking
How to be Artificially Intelligent
Robert Kowalski

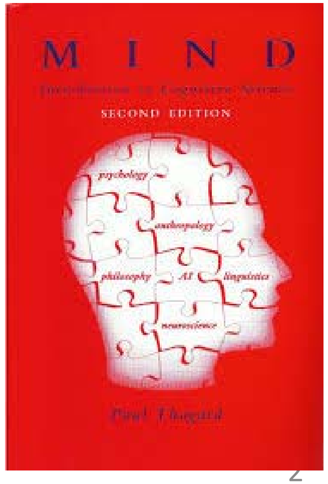

A Study in Scarlet
Sir Arthur Conan Doyle
A Famous Sherlock Holmes Story
4/-


International Bestseller
The Pyramid Principle
Barbara Minto
Logical writing, thinking and problem solving
FT Prentice Hall


'A lifetime's worth of wisdom'
Steven D. Levitt, co-author of Freakonomics
The International Bestseller
Thinking, Fast and Slow
Daniel Kahneman
Winner of the Nobel Prize


Thinking and Deciding
Fourth Edition
Jonathan Baron


Computer Science Essentials
Logic for Problem Solving, Revisited
Robert Kowalski
Edited by Thom Frühwirth


M I N D
Second Edition
psychology
anthropology
philosophy
AI
linguistics
neuroscience

# Conclusions

The task of an intelligent agent is to make its goals true in the world as seen through its beliefs.

# Conclusions

The task of an intelligent agent is to make its goals true in the world as seen through its beliefs.

- Goals
  - are more fundamental than beliefs
  - resemble production system rules
- Beliefs
  - have the form of logic programs

# Conclusions

The task of an intelligent agent is to make its goals true
in the world as seen through its beliefs.

- Goals
  - are more fundamental than beliefs
  - resemble production system rules
- Beliefs
  - have the form of logic programs
- Computational Logic (CL)
  - combines goal and beliefs
  - embeds abductive logic programming (ALP) in an agent cycle

# Outline of the talk

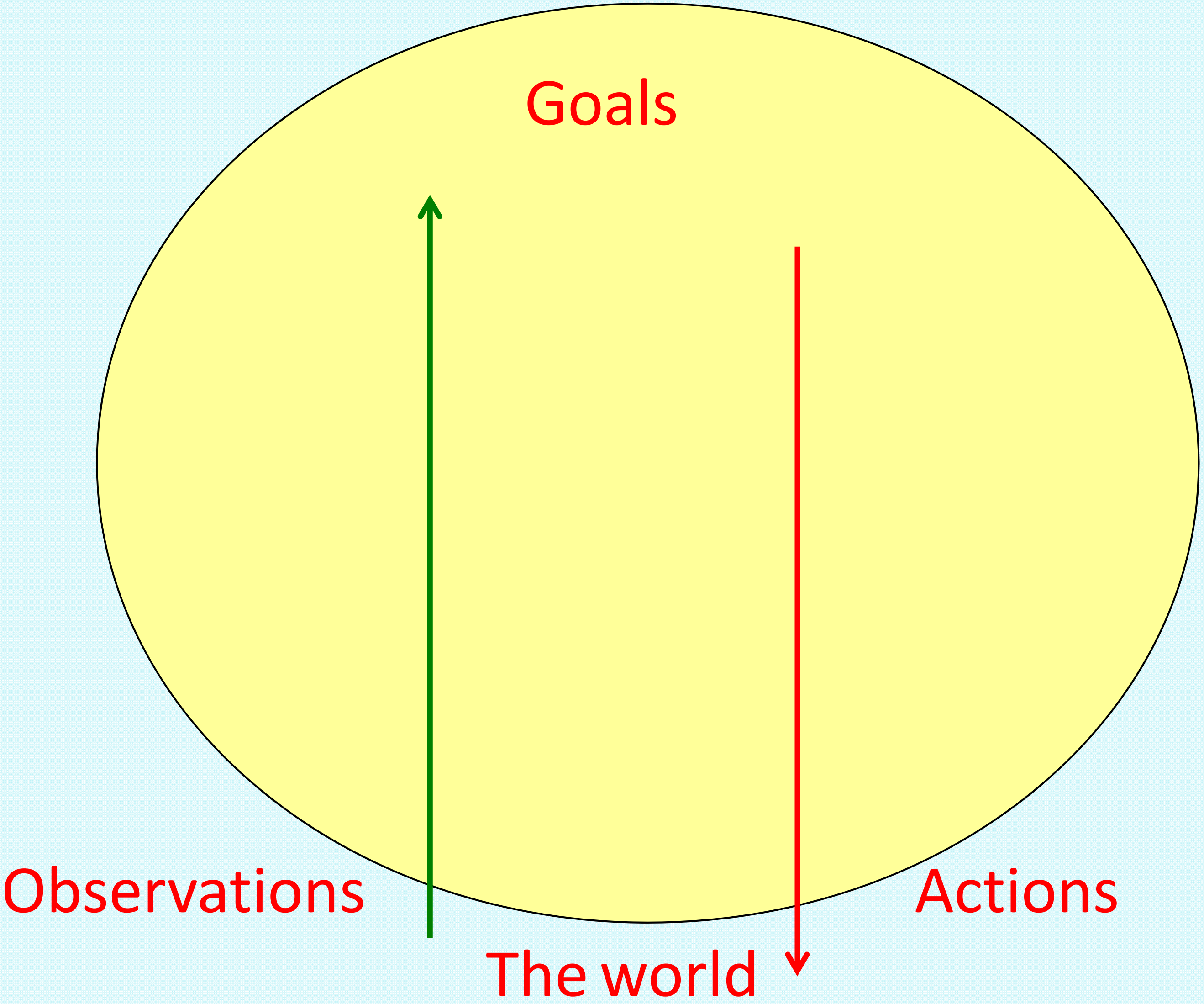
Overview

Logic programs represent beliefs

Production systems represent goals (but have no logic)

Computational Logic combines goals and beliefs embedded in an observe-think-decide-act agent cycle

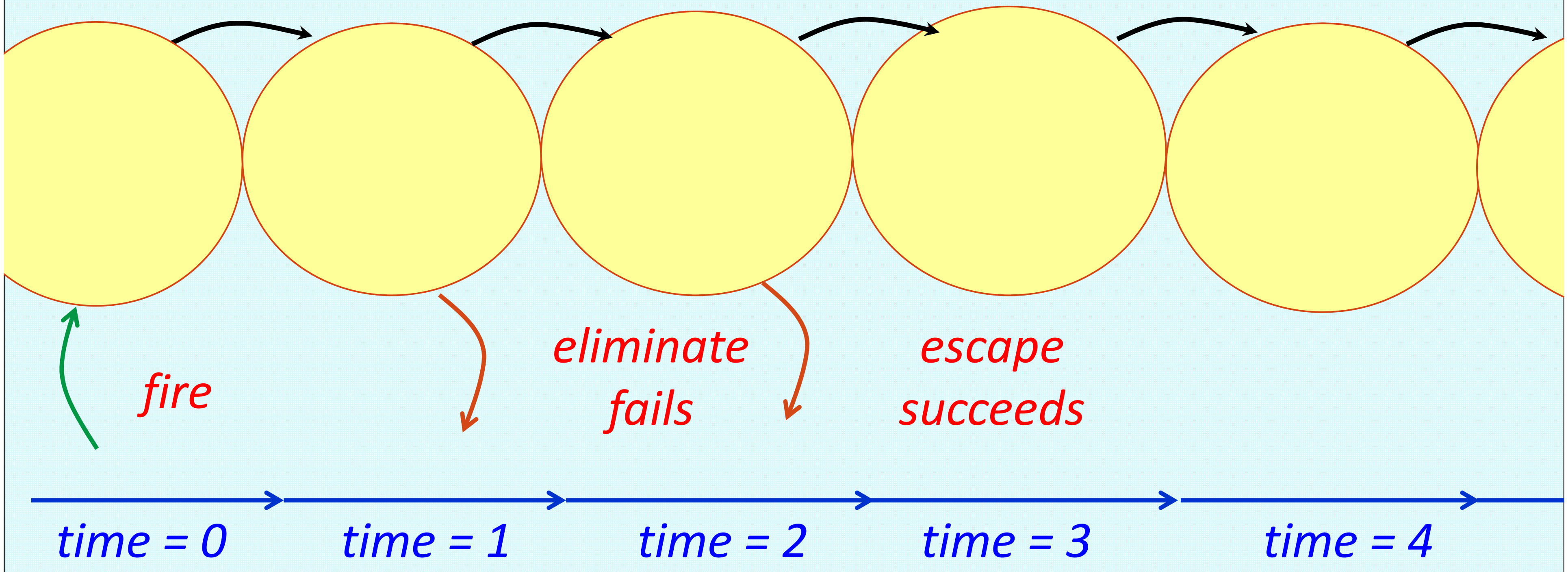
An agent's task in life is to perform actions to make its goals and observations true



Goal: *fire(T) → eliminate(T+1) ∨ escape(T+2)*

Observation: *fire(1)*

Action: *escape(3)*




Goal and observation are true in the model of the world described by

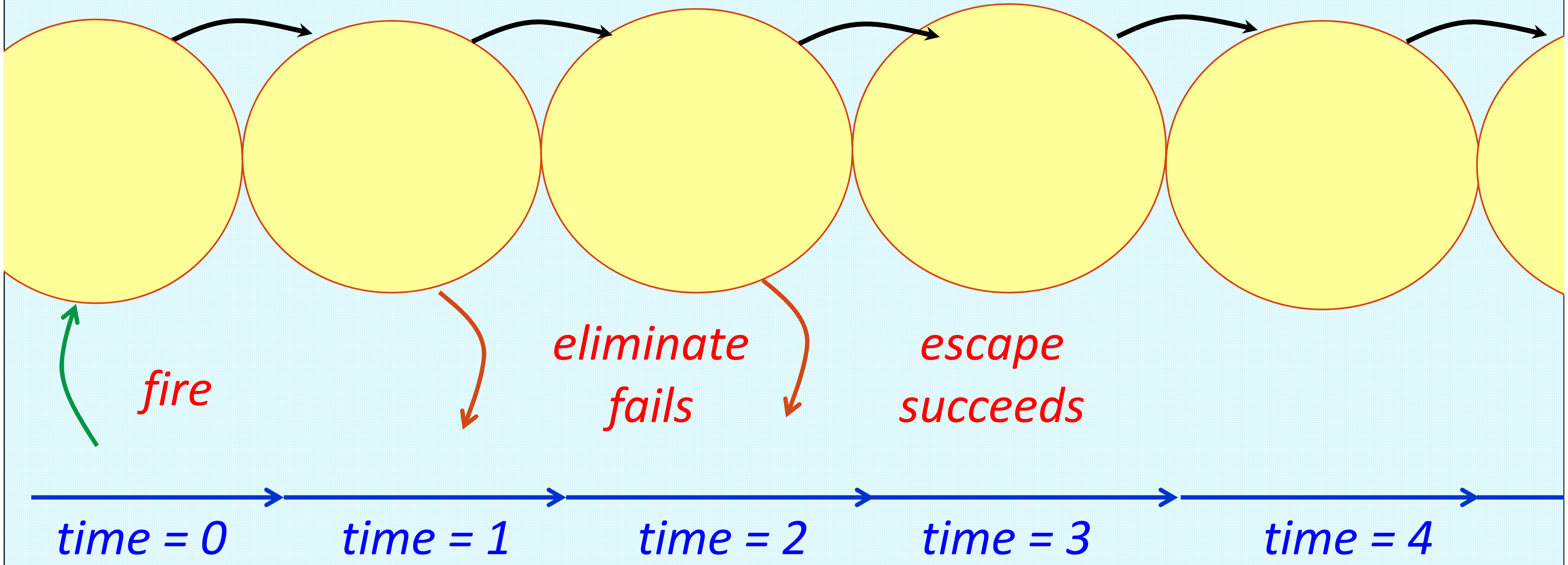
*{fire(1), escape(3)}*

Goal: *threat(T)* $\rightarrow$ *eliminate(T+1)* $\vee$ *escape(T+2)*

Belief: *threat(T)* $\leftarrow$ *fire(T)*

Observation: *fire(1)*

Action: *escape(3)*




Goal and observation are true in the model of the world described by

*{fire(1), threat(1), escape(3)}*

# The distinction between goals and beliefs is the foundation of SBVR

## SBVR – From Wikipedia:

"The Semantics of Business Vocabulary and Business Rules (SBVR) is an adopted standard of the Object Management Group (OMG) intended to be the basis for formal and detailed natural language declarative description of a complex entity, such as a business. "

# The distinction between goals and beliefs is the foundation of SBVR

## SBVR – From Wikipedia:

“The Semantics of Business Vocabulary and Business Rules (SBVR) is an adopted standard of the Object Management Group (OMG) intended to be the basis for formal and detailed natural language declarative description of a complex entity, such as a business. “

## From Baisley, Hall and Chapin:

“Distinguishing between
guidance (rules that people break) and
structural rules (rules about meaning)
is very important in understanding business rules.”

## SBVR - Example from Baisley, Hall and Chapin

It is obligatory that each person on a bus has a ticket.

*A person on a bus either has a ticket or is breaking the rule.*

It is logically necessary that each person on a bus has a ticket.

*Being on a bus implies that there is a ticket.*

These modalities are not nested as in normal modal logic.

# SBVR - Example from Baisley, Hall and Chapin

It is obligatory that each person on a bus has a ticket.

*A person on a bus either has a ticket or is breaking the rule.*

It is logically necessary that each person on a bus has a ticket.

*Being on a bus implies that there is a ticket.*

These modalities are not nested as in normal modal logic.

Goal: *a person is on a bus* $\rightarrow$ *the person has a ticket.*

Belief: *a person is on a bus* $\rightarrow$ *the person has a ticket.*

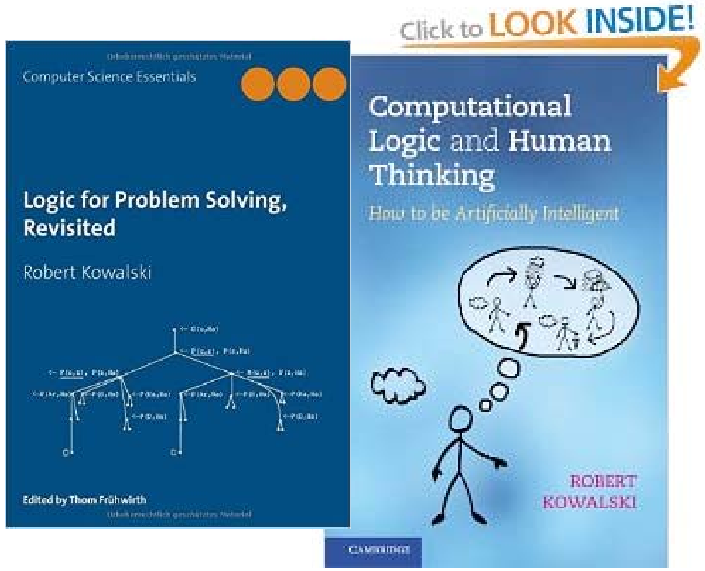
The distinction between goals and beliefs
is fundamental in database systems

Datalog rules = beliefs
*manager(X) → employee(X)*

Integrity constraints = goals
*manager(X) → ∃Y supervises(X, Y )*
*employee(X), employer(X) → false*
*supervises(X, Y ), supervises(X′, Y ) → X = X′*

From Datalog± (including ontologies and integrity constraints; Cali, Gottlob, Lukasziewicz; 2009)

Abductive logic programming (ALP) combines goals (integrity constraints) and beliefs (logic programs)

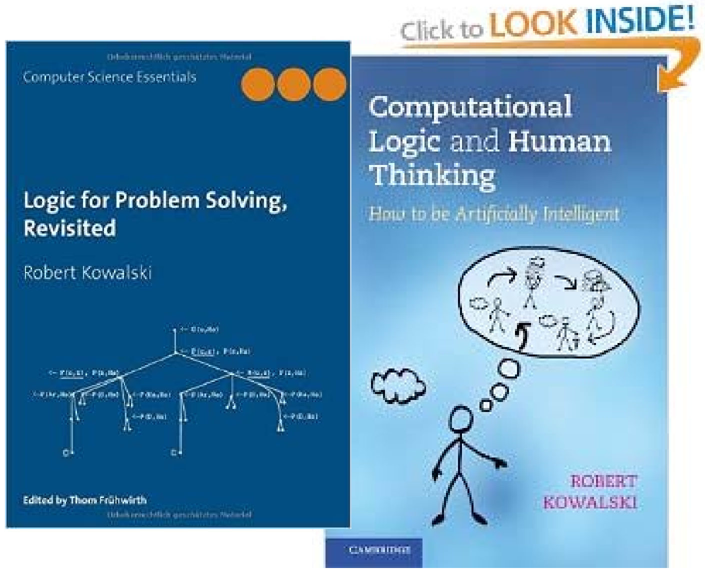



Beliefs:

$$conclusion\ if\ condition_1\ and\ ....\ and\ condition_n$$

or: $\forall X\ [condition_1 \wedge .... \wedge condition_n \rightarrow conclusion]$

Abductive logic programming (ALP) combines goals (integrity constraints) and beliefs (logic programs)




Beliefs:

*conclusion if condition*$_1$ *and .... and condition*$_n$

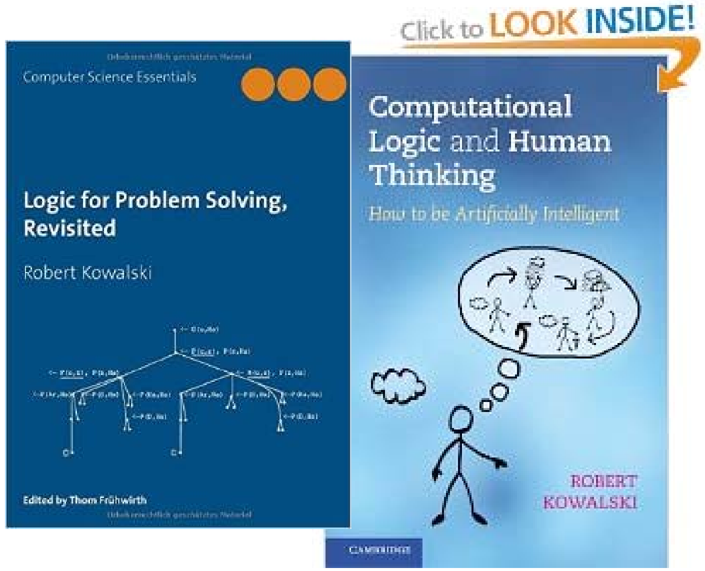
or: $\forall X\ [condition_1 \wedge .... \wedge condition_n \rightarrow conclusion]$

Maintenance goals:

*If condition*$_1$ *and .... and condition*$_n$
*then conclusion*$_1$ *or .... or conclusion*$_m$

or: $\forall X\ [condition_1 \wedge .... \wedge condition_n$
$\rightarrow \exists Y\ [conclusion_1\ or\ ....\ or\ conclusion_m]$

Abductive logic programming (ALP) combines goals (integrity constraints) and beliefs (logic programs)




Beliefs:

*conclusion if condition$_1$ and …. and condition$_n$*

or: $\forall X\ [condition_1 \wedge .... \wedge condition_n \rightarrow conclusion]$

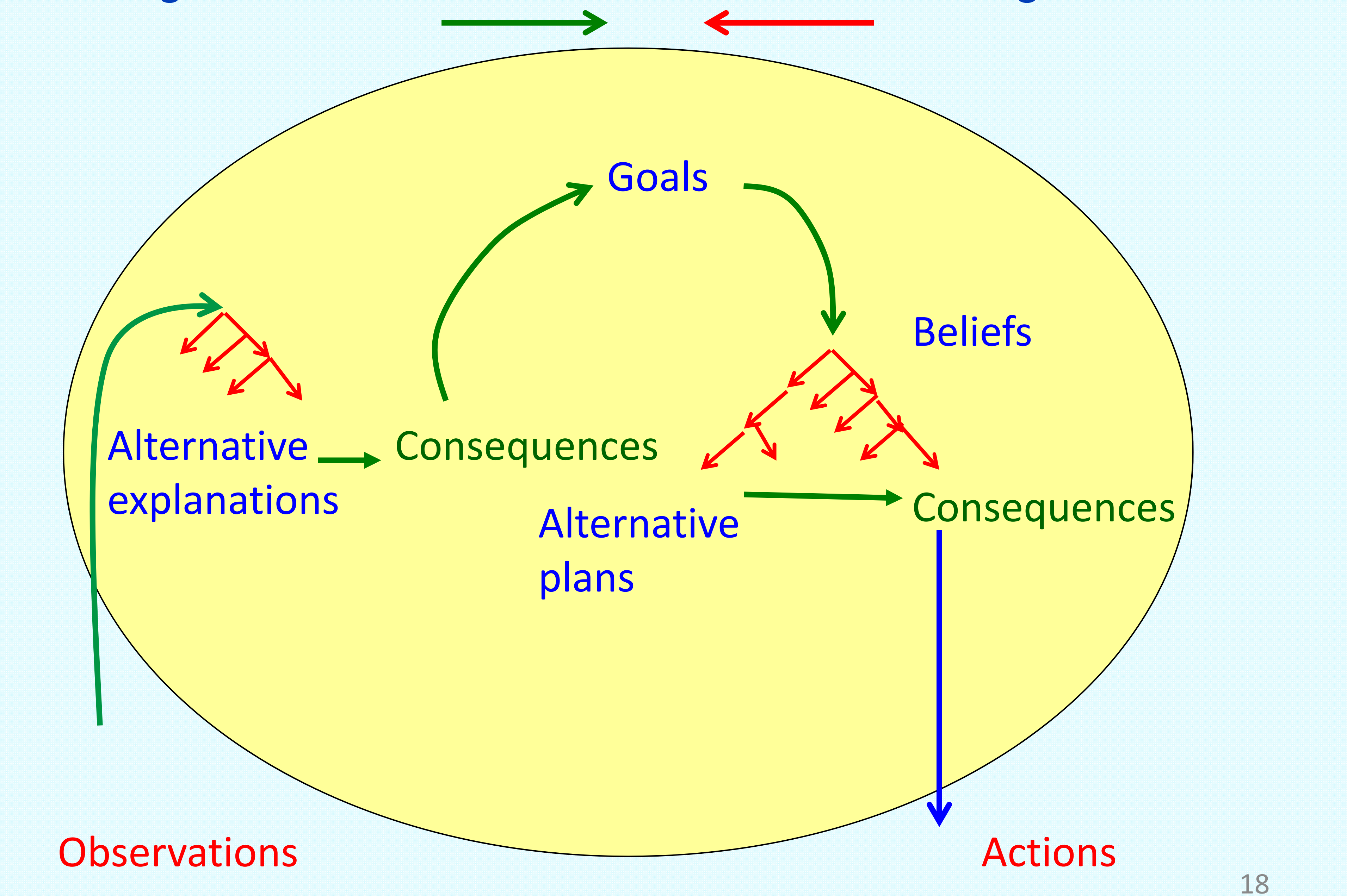
Maintenance goals:

*If condition$_1$ and …. and condition$_n$*
*then conclusion$_1$ or …. or conclusion$_m$*

or: $\forall X\ [condition_1 \wedge .... \wedge condition_n$
$\rightarrow\ \exists\ Y\ [conclusion_1\ or\ ....\ or\ conclusion_m]$

It can be hard to tell the difference.

# ALP agents combine forward and backward reasoning



# Forward and backward reasoning

**Forward reasoning**

Given *A, B* and *if A and B then C* derive *C*.

**Backward reasoning**

Given goal *C?* and *C if A and B* derive subgoals *A and B?*.

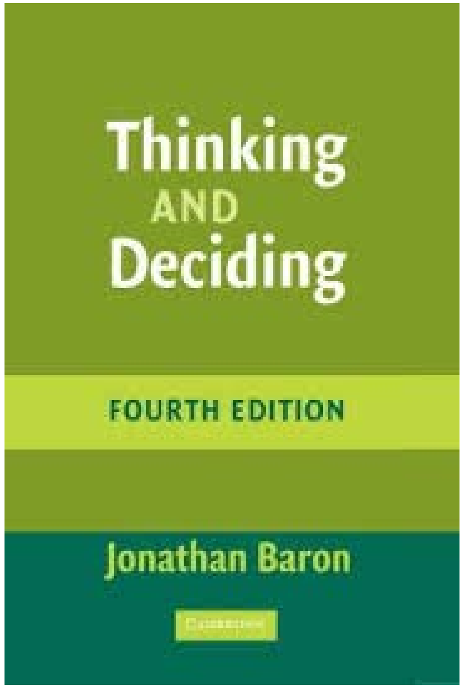
# Forward reasoning

Forward reasoning derives consequences from assumption.

But also derives achievement goals from maintenance goals:

Given beliefs *A, B*
and maintenance goal *if A and B then C* !
derive achievement goal *C* !

# Jonathan Baron “Thinking and Deciding” (Fourth edition, 2008)

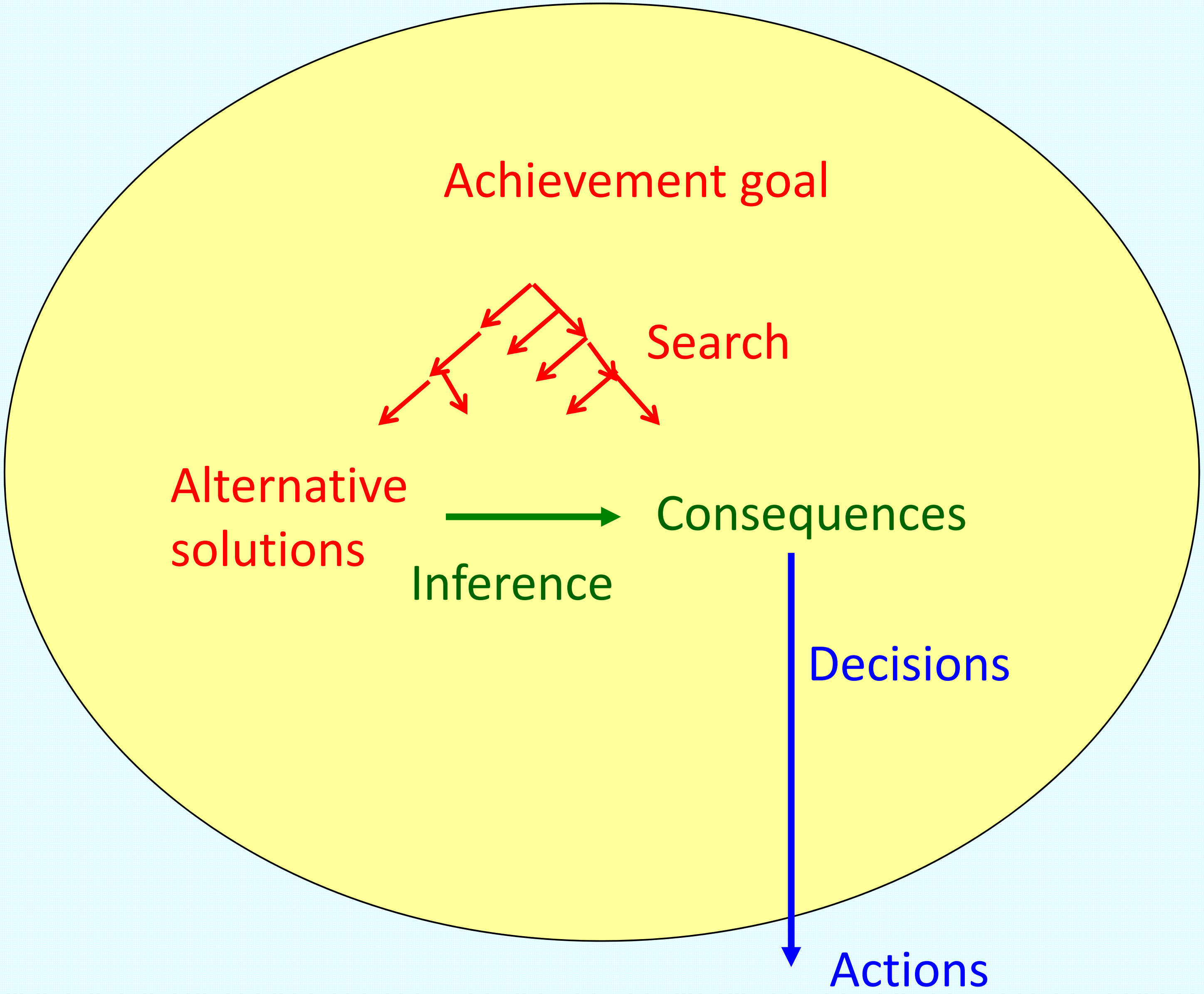



“*Thinking* about actions,
beliefs and personal goals
can all be described in terms of
a common framework,

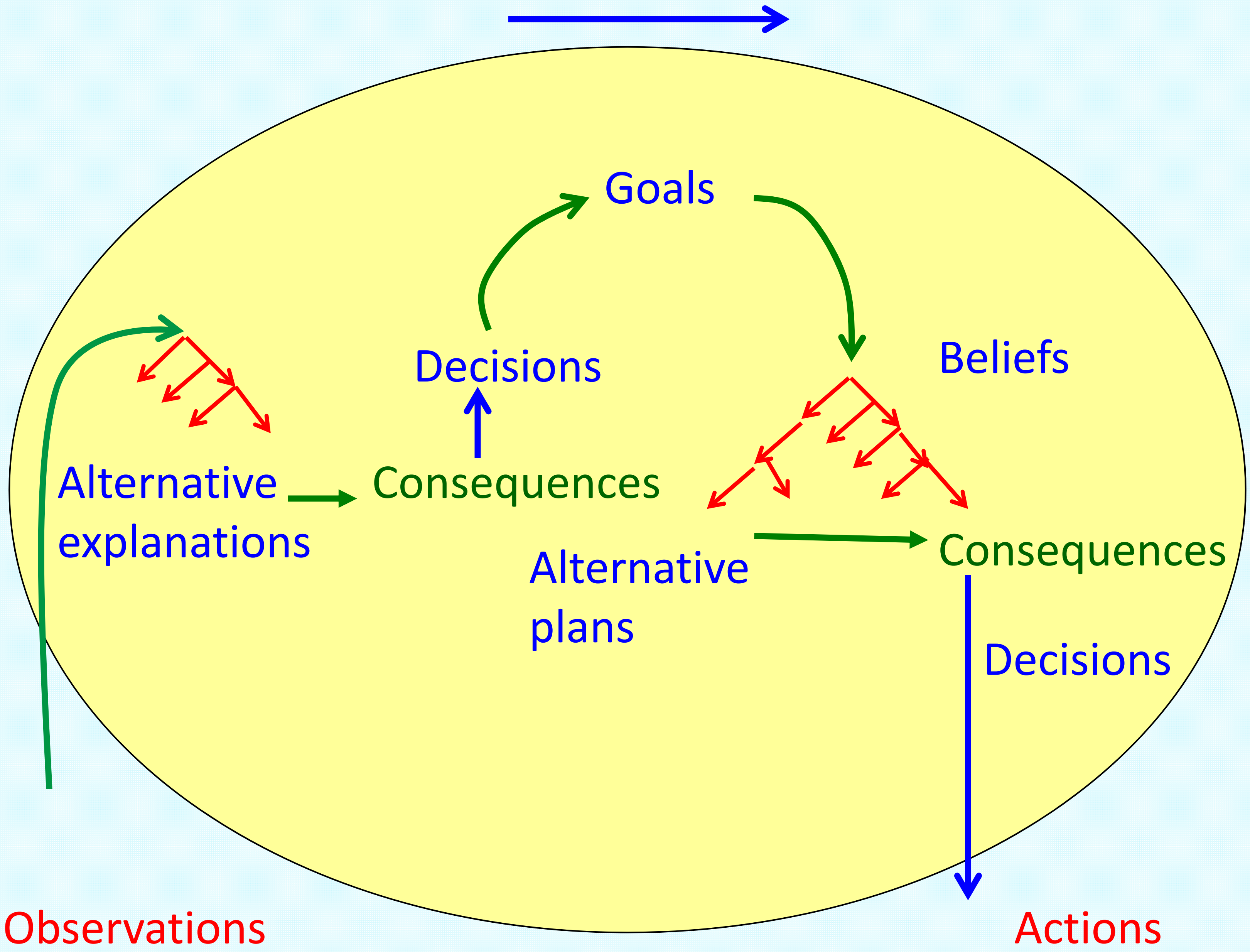
which asserts that
thinking consists of *search* and *inference*.

We *search* for certain objects and then *make inferences*
from and about the objects we have found.” (page 6)

# Baron's view of thinking and deciding



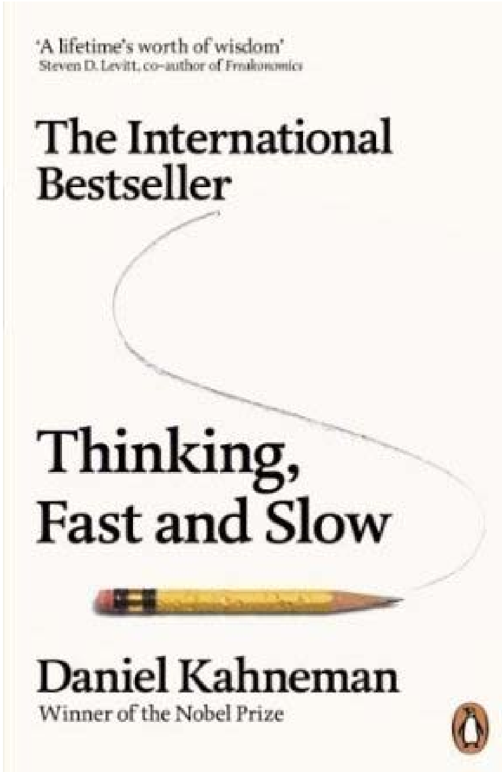
ALP agents need to make decisions



# The dual process model combines two systems of thinking

System 1 operates automatically and quickly, with little or no effort and no sense of voluntary control.

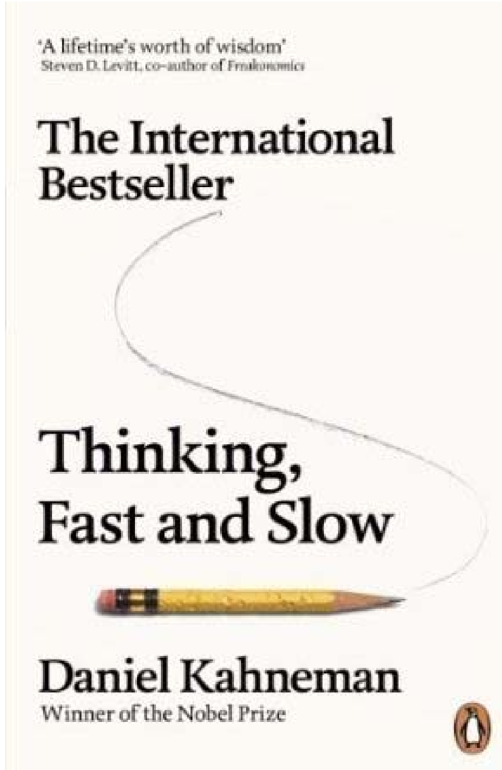
System 2 allocates attention to the effortful mental activities that demand it, including complex computations.



## The dual process model

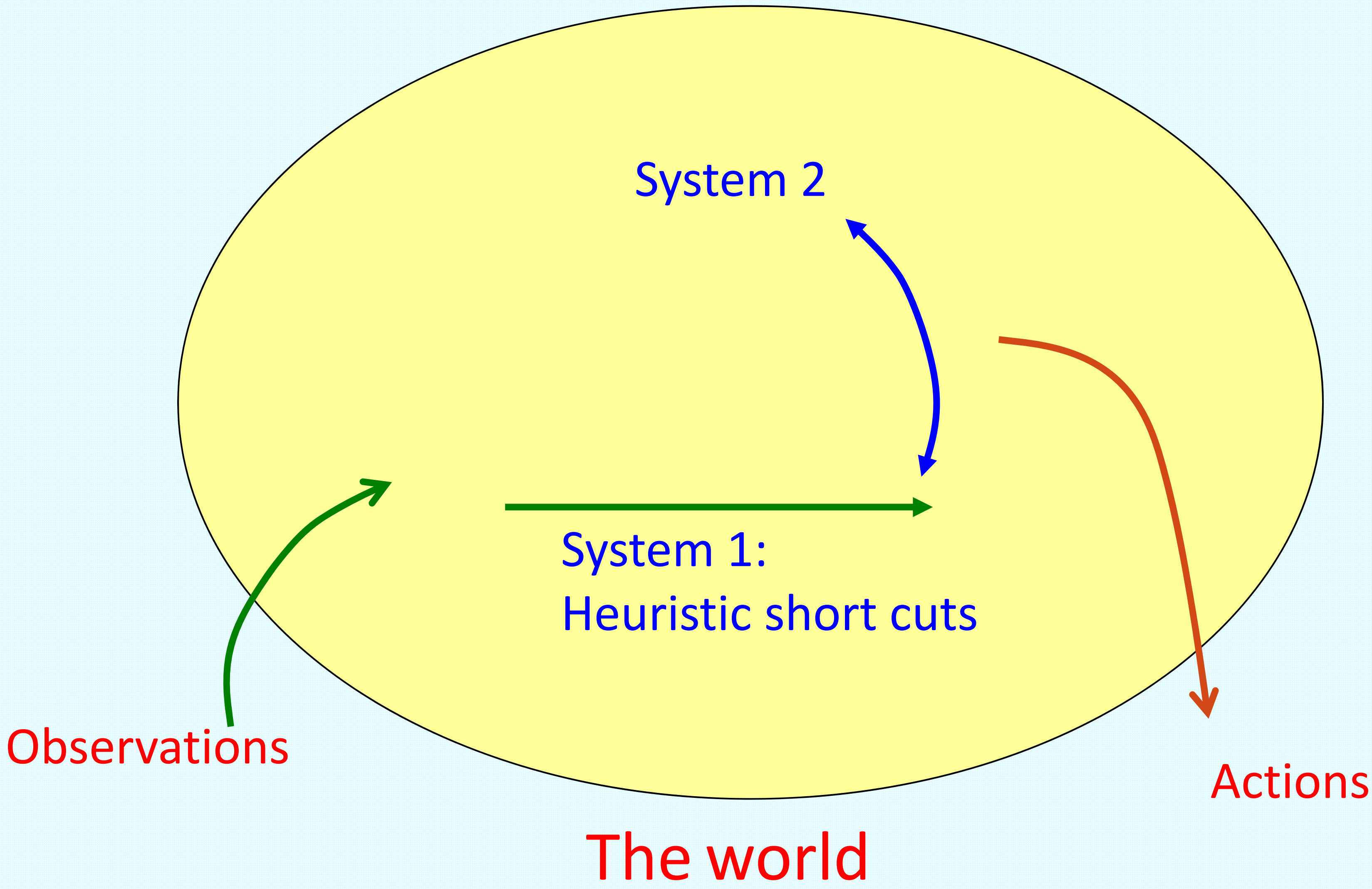
System 1 “quickly proposes intuitive answers to judgement problems as they arise”,

System 2 “monitors the quality of these proposals, which it may endorse, correct, or override”.

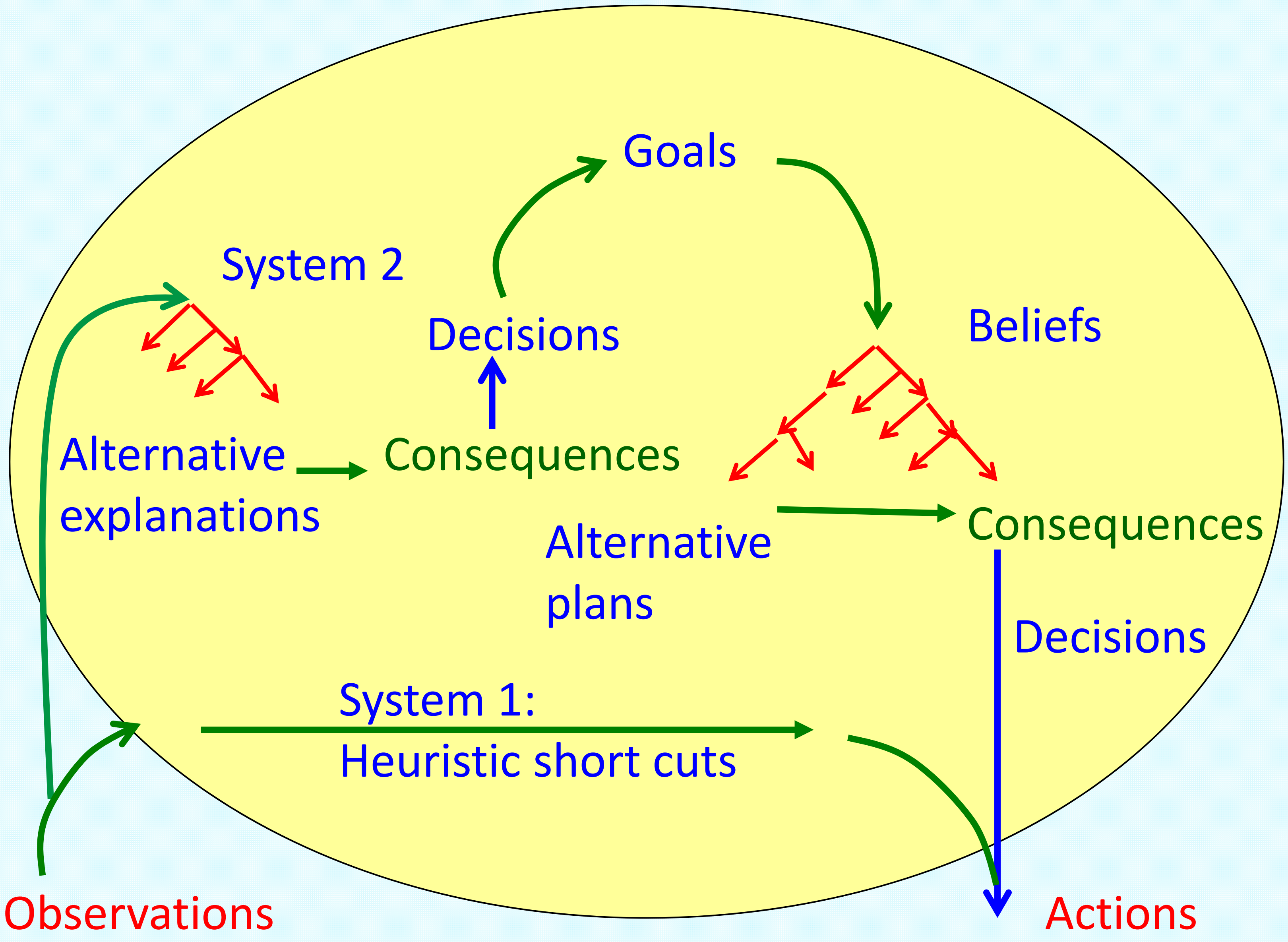



System 2 is activated when an event is detected that violates the model of the world that system 1 maintains.

# The Dual Process Model



# ALP agents (CL) as a unifying framework



# Outline of the talk

Overview

Logic programs represent beliefs

Production systems represent goals (but have no logic)

Computational Logic combines goals and beliefs embedded in an observe-think-decide-act agent cycle

# The London underground emergency notice as a logic program

## Emergencies

Press the alarm signal button to alert the driver.

The driver will stop
if any part of the train is in a station.

If not, the train will continue to the next station,
where help can more easily be given.

There is a 50 pound penalty for improper use.

# The hidden logic (+ control) of the Emergency Notice

Reason backwards to reduce goals to subgoals:

*the driver is alerted*
*if you press the alarm signal button.*

# The hidden logic (+ control) of the Emergency Notice

Reason forwards to derive possible consequences of actions:

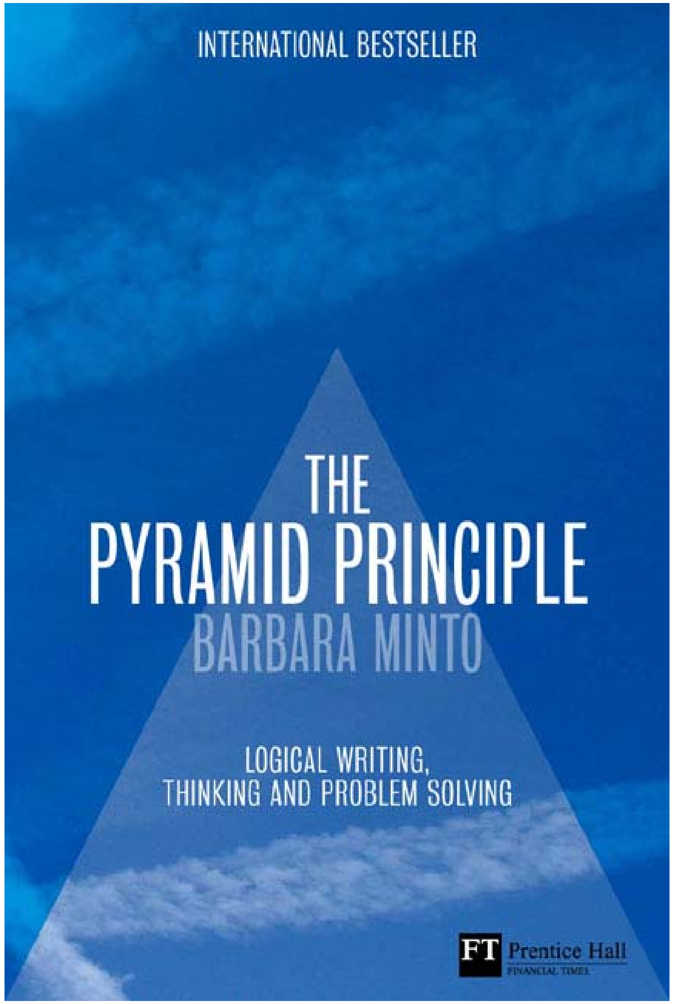
*the driver will stop the train in a station*
*if the driver is alerted*
*and any part of the train is in the station.*

*the driver will stop the train in the next station*
*if the driver is alerted*
*and not any part of the train is in a station.*

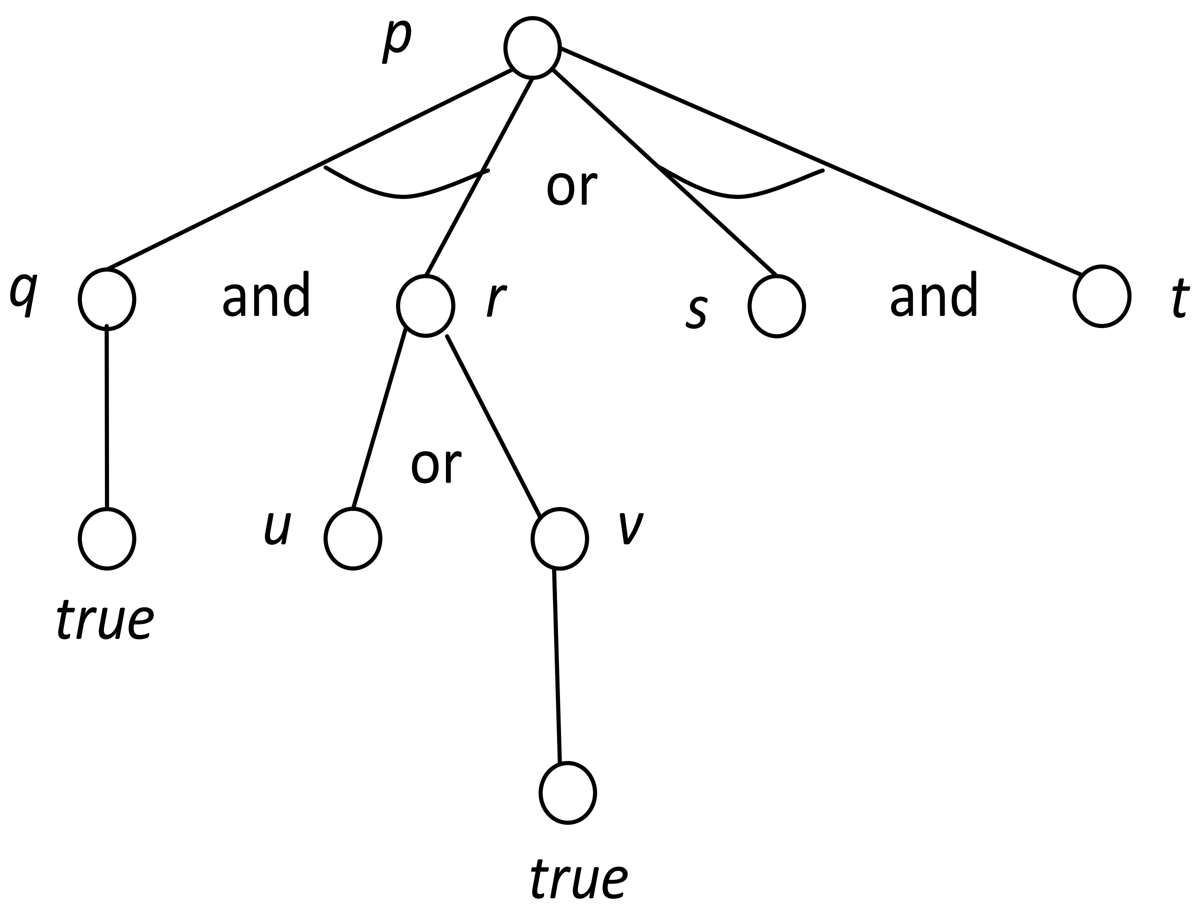
*help can more easily be given in an emergency*
*if the train is in a station.*

*You may be liable to a £50 penalty*
*if you use the alarm signal button improperly*

## Backward reasoning as a guide for clear thinking, writing and problem solving



Backward reasoning generates
a pyramid (or triangle or and-or tree)




$\leftarrow p$

$p \leftarrow q \wedge r$

$p \leftarrow s \wedge t$

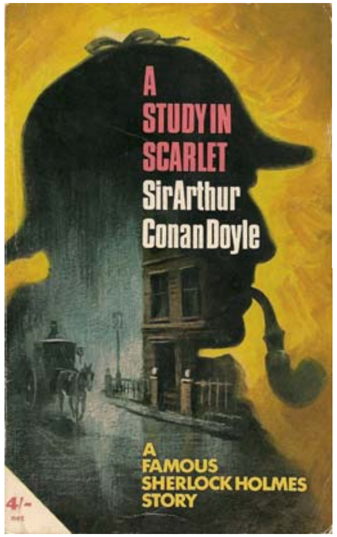
$r \leftarrow u$

$r \leftarrow v$

$q$

$v$

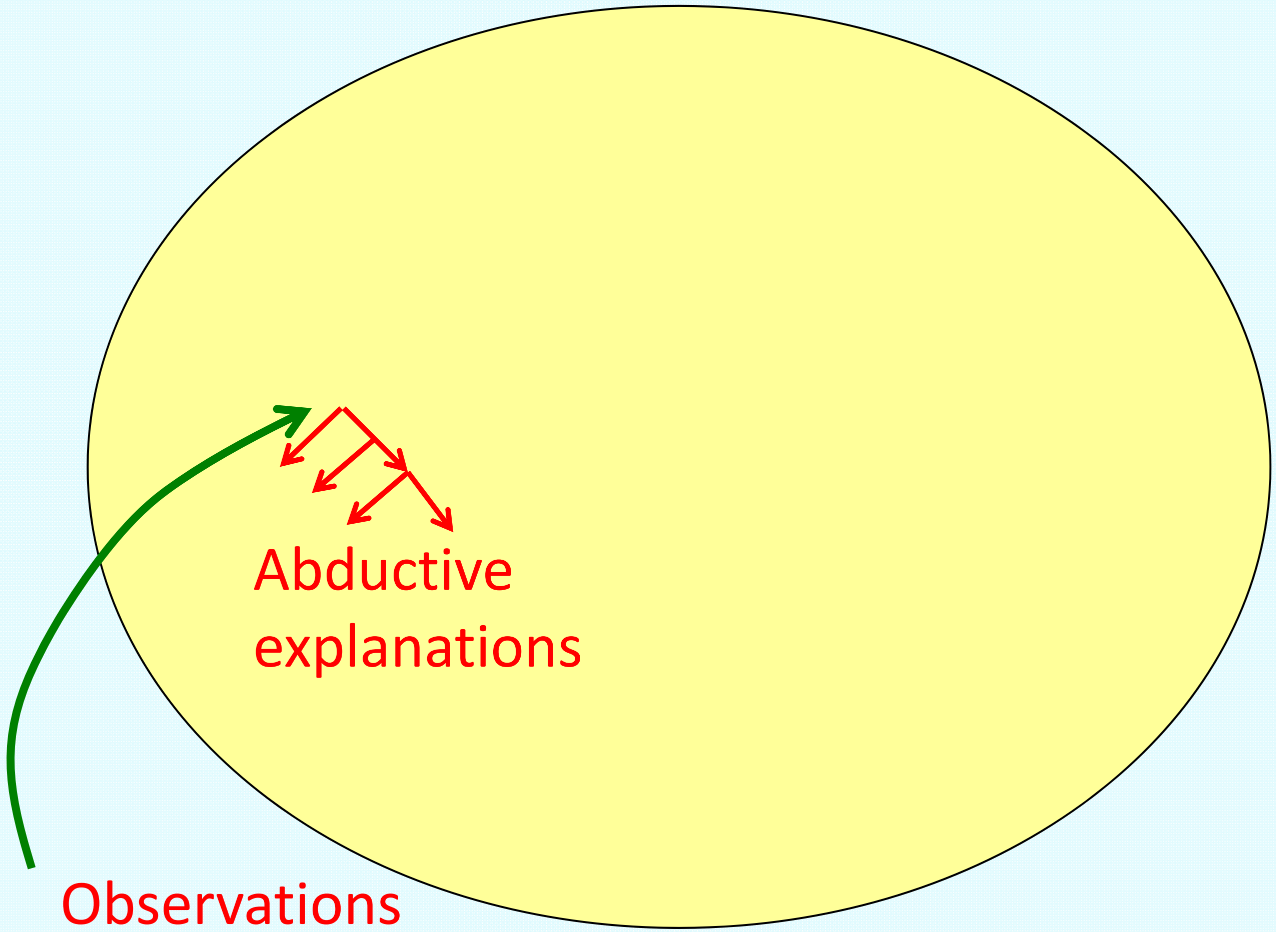
As Sherlock Holmes explained to Dr. Watson, in *A Study in Scarlet*:




"In solving a problem of this sort,
the grand thing is to be able to reason backward.
That is a very useful accomplishment,
and a very easy one,
but people do not practise it much.

In the everyday affairs of life,
it is more useful to reason forward,
and so the other comes to be neglected.
There are fifty who can reason synthetically
for one who can reason analytically."

Sherlock Holmes used backward reasoning to generate hypotheses to explain observations. This is called abduction. He called it “deduction”.



Goals in logic programming are restricted to achievement goals

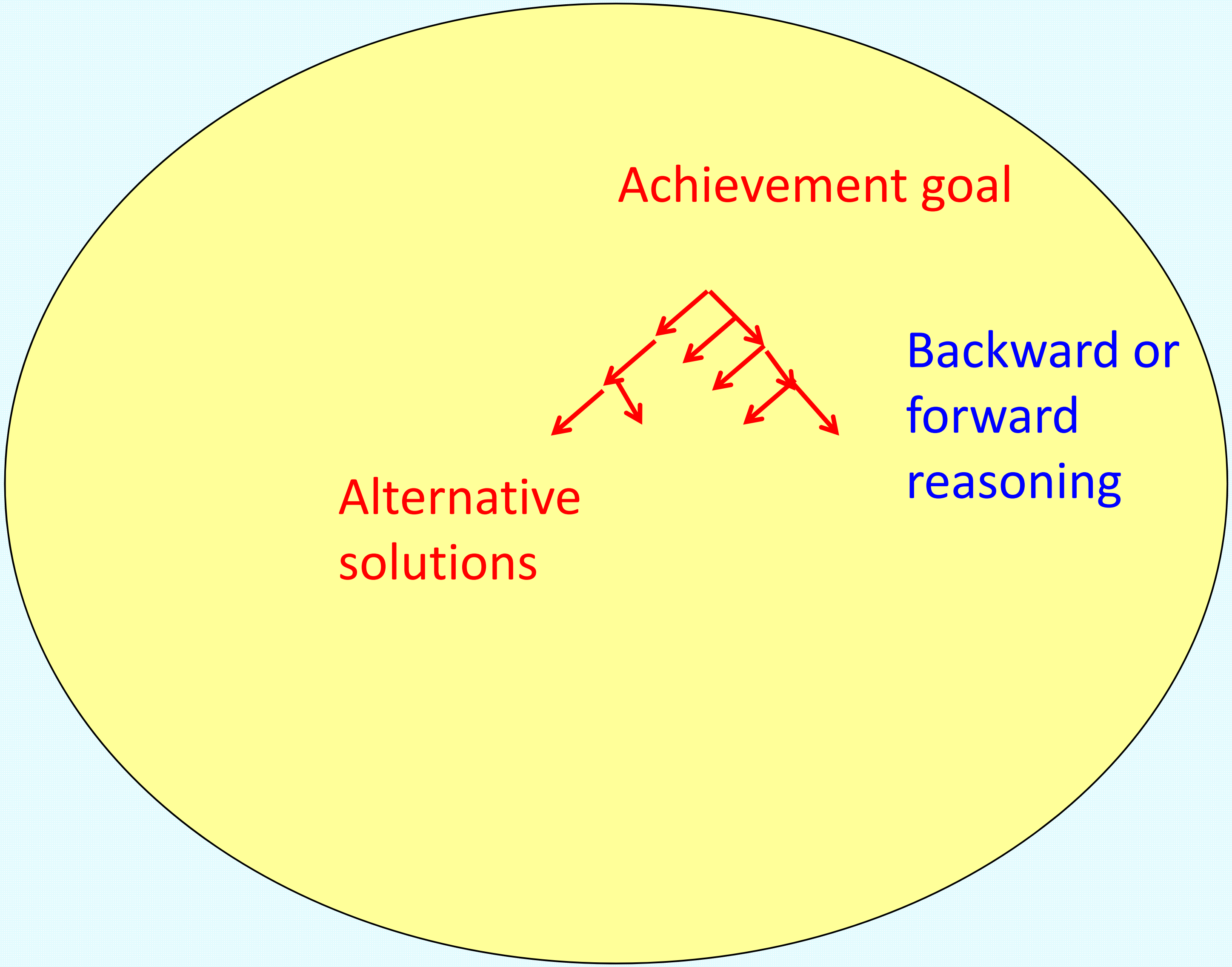
$condition_1 \wedge condition_2$ .... $\wedge$ $condition_n$ ?

where $condition_1$ and $condition_2$ .... and $condition_n$ are atomic formulas or negations of atomic formulas.

Variables *X* represent values that need to be found.

$\exists$ *X* [$condition_1 \wedge condition_2$ .... $\wedge$ $condition_n$]?

# The logic programming view of thinking



# Outline of the talk

Overview

Logic programs represent beliefs

Production systems represent goals (but have no logic)

Computational Logic combines goals and beliefs
embedded in an observe-think-decide-act agent cycle

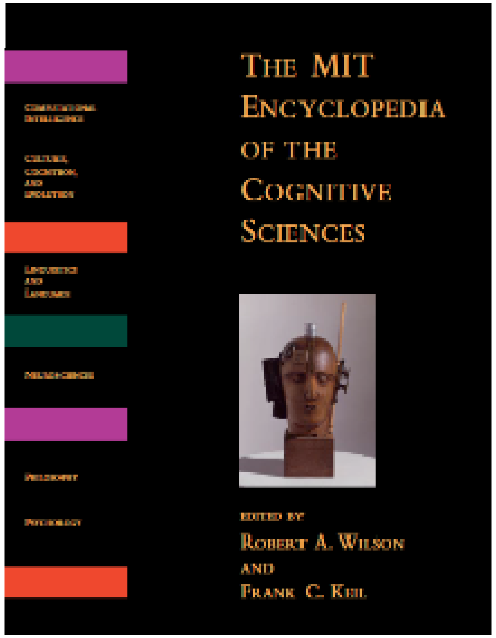
## Production Systems —*Herbert A. Simon*

Production systems are computer languages that are widely employed for representing the processes that operate in models of cognitive systems (NEWELL and Simon 1972).

In a production system, all of the instructions (called productions) take the form:

IF<<conditions>, THEN<<actions>,

That is to say, "if certain conditions are satisfied, then take the specified actions" (abbreviated C → A). Production system languages have great generality: they can possess the full power and generality of a Turing machine (see TURING). They have an obvious affinity to the classical stimulus-response (S → R) connections in psychology, but greater complexity and flexibility, for, in production systems, both



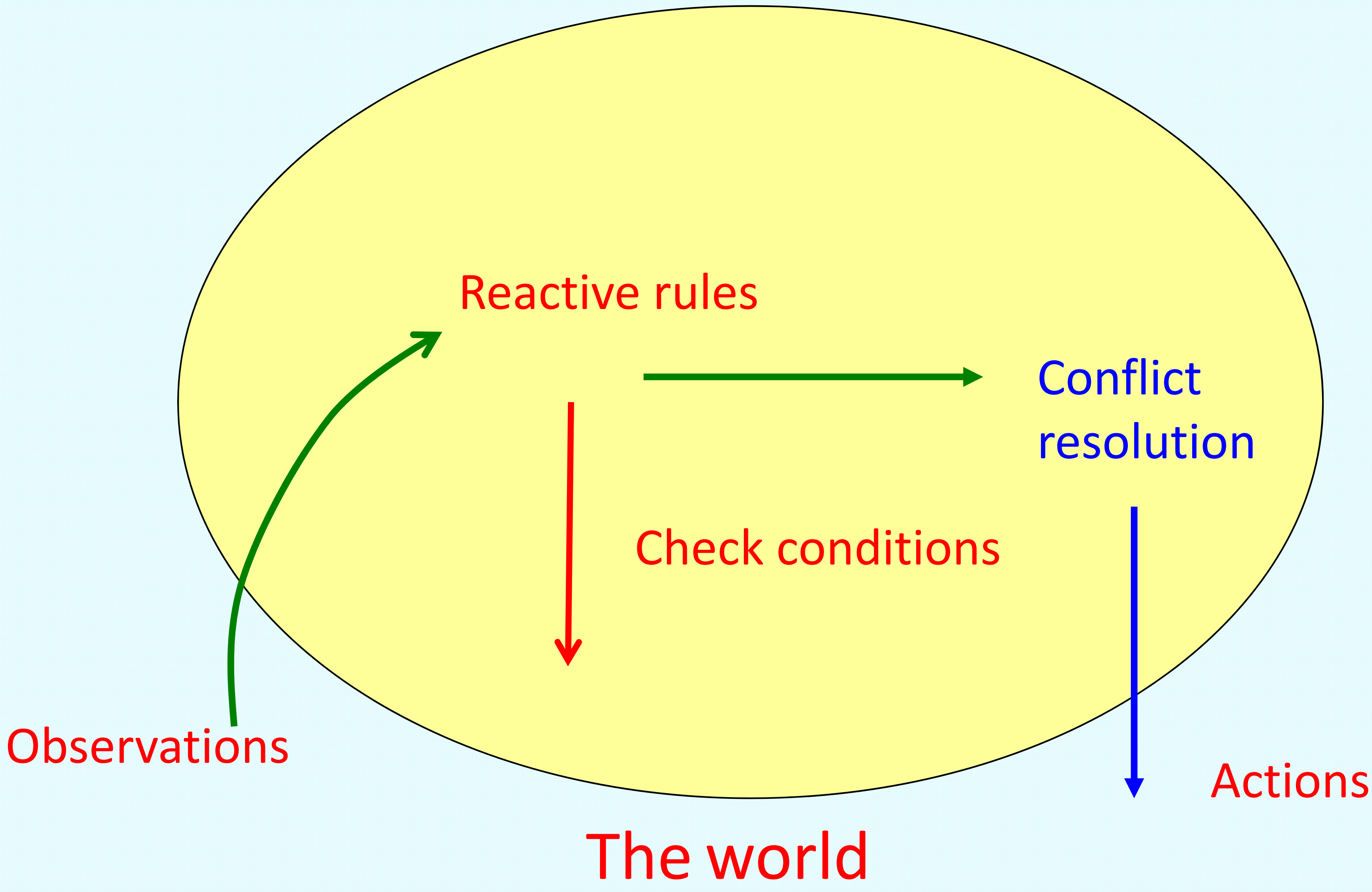
Production rules implement reactive rules, which are a kind of maintenance goals.

$$threat \rightarrow eliminate$$
$$threat \rightarrow escape$$

Production systems use "conflict resolution" to decide between conflicting actions. (They confuse "and" and "or".)

In production rules, $\rightarrow$ does not mean logical if-then.
Change of state is implicit.

# The Production System view of thinking



# Three kinds of production rules

- Reactive rules (or maintenance goals):

$$threat \rightarrow eliminate$$
$$threat \rightarrow escape$$

# Three kinds of production rules

- Reactive rules (or maintenance goals):

  $$threat \rightarrow eliminate$$
  $$threat \rightarrow escape$$

- Logic programs (or beliefs) executed forward:

  $$fire \rightarrow threat$$

# Three kinds of production rules

- Reactive rules (or maintenance goals):

  *threat* $\rightarrow$ *eliminate*

  *threat* $\rightarrow$ *escape*

- Logic programs (or beliefs) executed forward:

  *fire* $\rightarrow$ *threat*

- Logic programs (or beliefs) executed backwards, simulated by forward chaining:

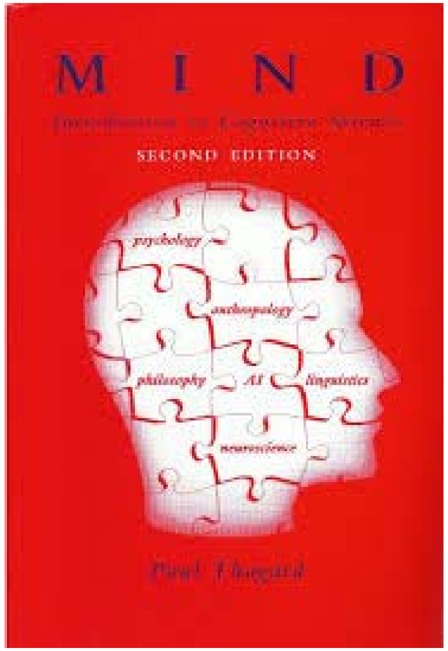
  *eliminate* $\rightarrow$ *get help*

  *get help* $\rightarrow$ *press the alarm*

Many authors are confused about the relationship between logic and production systems.

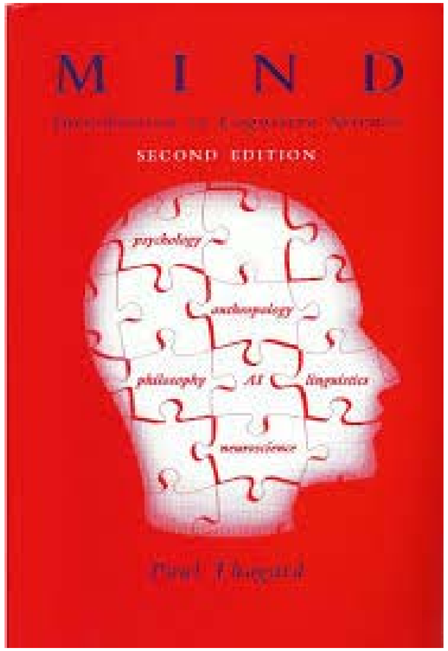
“Unlike logic, rule-based systems can easily represent strategic information about what to do”:

IF you want to go home
AND you have the bus fare,
THEN you can catch a bus.



Many authors are confused about the relationship between logic and production systems.

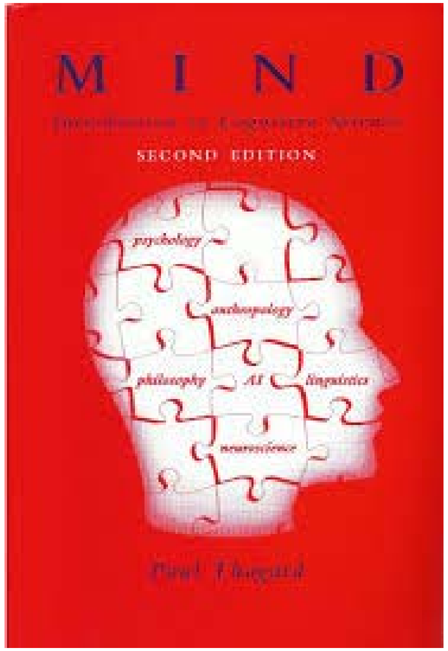
"Unlike logic, rule-based systems can easily represent strategic information about what to do":




IF you want to go home
AND you have the bus fare,
THEN you can catch a bus.

Logic program: *you go home*
*if you have the bus fare,*
*and you catch a bus.*

## Many authors are confused about the relationship between deduction and search




"In logic-based systems
the fundamental operation of thinking
is logical deduction,
but from the perspective of rule-based systems
the fundamental operation of thinking is search."

## The relationship between deduction and search

IF you drive on highway 1,
THEN you can get from university city to home city.

IF you take the parkway,
THEN you can get from university city to the highway.

IF you take a bus from the bus depot,
THEN you can get from university city to home city.
etc.

# The relationship between deduction and search

IF you drive on highway 1,
THEN you can get from university city to home city.

IF you take the parkway,
THEN you can get from university city to the highway.

IF you take a bus from the bus depot,
THEN you can get from university city to home city.
etc.

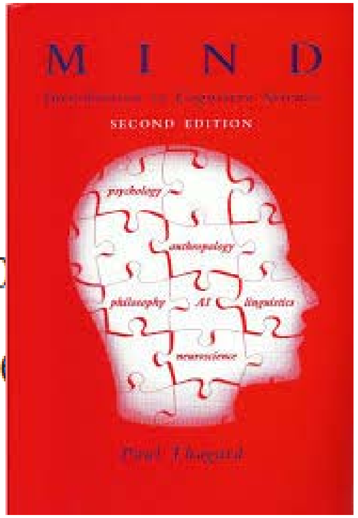
Logic program:

*you can get from A to B*
*if there is a Road from A to B and you drive on the Road*

*you can get from A to B*
*if there is a Bus from A to B and you take the Bus.*

*there is highway 1 from university city to home city.*
*etc.*

Many authors are confused about the relationship between logic and default reasoning.




systems. But the developers of rule-based systems have been happ
some of the representational rigor of logic-based systems for th
increased computational power. One advantage comes from the
rules do not have to be interpreted as universally true. The logical generalization (for all *x*) (student (*x*) → overworked (*x*)) must be interpreted as saying that every student is overworked. But the rule that *IF x is a student THEN x is overworked* can be interpreted as a *default*, that is, as a rough generalization that can admit exceptions. We might have another rule that says that *IF x is a student and x is taking only easy courses, THEN x is not overworked*. These two rules might coexist in the same system, but the result

# Logic programs can represent default reasoning.

IF x is a student, THEN x is overworked.

IF x is a student AND x is taking only easy courses, THEN x is not overworked.

# Logic programs can represent default reasoning.

IF x is a student, THEN x is overworked.

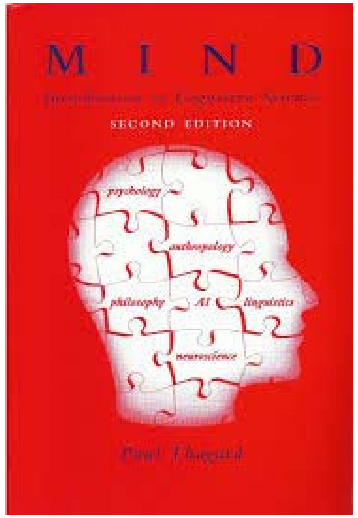
IF x is a student AND x is taking only easy courses, THEN x is not overworked.

*X is overworked*
*if X is a student*
*and not X is taking only easy courses.*

Logic programs can have negative conditions, intrepreted as negation as failure.

# Many authors are confused about the relationship between deduction and search

Most of Thagard’s examples of rules are examples of logic programs.



# AgentSpeak(L): BDI Agents speak out in a logical computable language

Anand S. Rao

**Abstract**

Belief-Desire-Intention (BDI) agents have been investigated by many researchers from both a theoretical specification perspective and a practical design perspective. However, there still remains a large gap between theory and practice. The main reason for this has been the complexity of theorem-proving or model-checking in these expressive specification logics. Hence, the implemented BDI systems have tended to use the three major attitudes as data structures, rather than as modal operators. In this paper, we provide an alternative formalization of BDI agents by providing an operational and proof-theoretic semantics of a language AgentSpeak(L). This language can be viewed as an abstraction of one of the implemented BDI systems (i.e., PRS) and allows agent programs to be written and interpreted in a manner similar to that of horn-clause logic programs. We

# AgentSpeak(L):

**Definition 5** If $e$ is a triggering event, $b_1$,...,$b_m$ are belief literals, and $h_1$,...,$h_n$ are goals or actions then $e$:$b_1 \wedge \ldots \wedge b_m \leftarrow h_1$;...;$h_n$ is a *plan*. The expression to the left of the arrow is referred to as the *head* of the plan and the expression to the right of the arrow is referred to as the *body* of the plan. The expression to the right of the colon in the head of a plan is referred to as the *context*. For convenience, we shall rewrite an empty body with the expression *true*.

With this we complete the specification of an agent. In summary, a designer specifies an agent by writing a set of base beliefs and a set of plans. This is similar to a logic programming specification of facts and rules. However, some of the major differences between a logic

**AgentSpeak(L):**

```
+location(waste,X):location(robot,X) &
                   location(bin,Y)
                   <- pick(waste);
                      !location(robot,Y);
                      drop(waste).
```

**AgentSpeak(L):**

```
+location(waste,X):location(robot,X) &
                   location(bin,Y)
                   <- pick(waste);
                      !location(robot,Y);
                      drop(waste).
```

Maintenance goal  in logical form with explicit time:

*location(waste, X, T1)* $\wedge$ *location(robot, X, T1)* $\wedge$
*location(bin, Y, T1)*
$\rightarrow$ *pick(waste, T1 +1)* $\wedge$
*reach(robot, Y, T2)* $\wedge$
*drop( waste, T2+1)*

Notice that <- is opposite to the logical reading.

# Two kinds of BDI rules

- Logic programs (or beliefs) executed backwards, simulated by forward chaining:

  *goal* *<-* *sub-goals and actions*
  (i.e. *goal* $\leftarrow$ *sub-goals and actions*)

# Two kinds of BDI rules

- Logic programs (or beliefs) executed backwards, simulated by forward chaining:

  *goal <- sub-goals and actions*
  (i.e. *goal* $\leftarrow$ *sub-goals and actions*)

- Reactive rules (or maintenance goals):

  *event and conditions <- goals and actions*
  (meaning *event and conditions* $\rightarrow$ *goals and actions*)

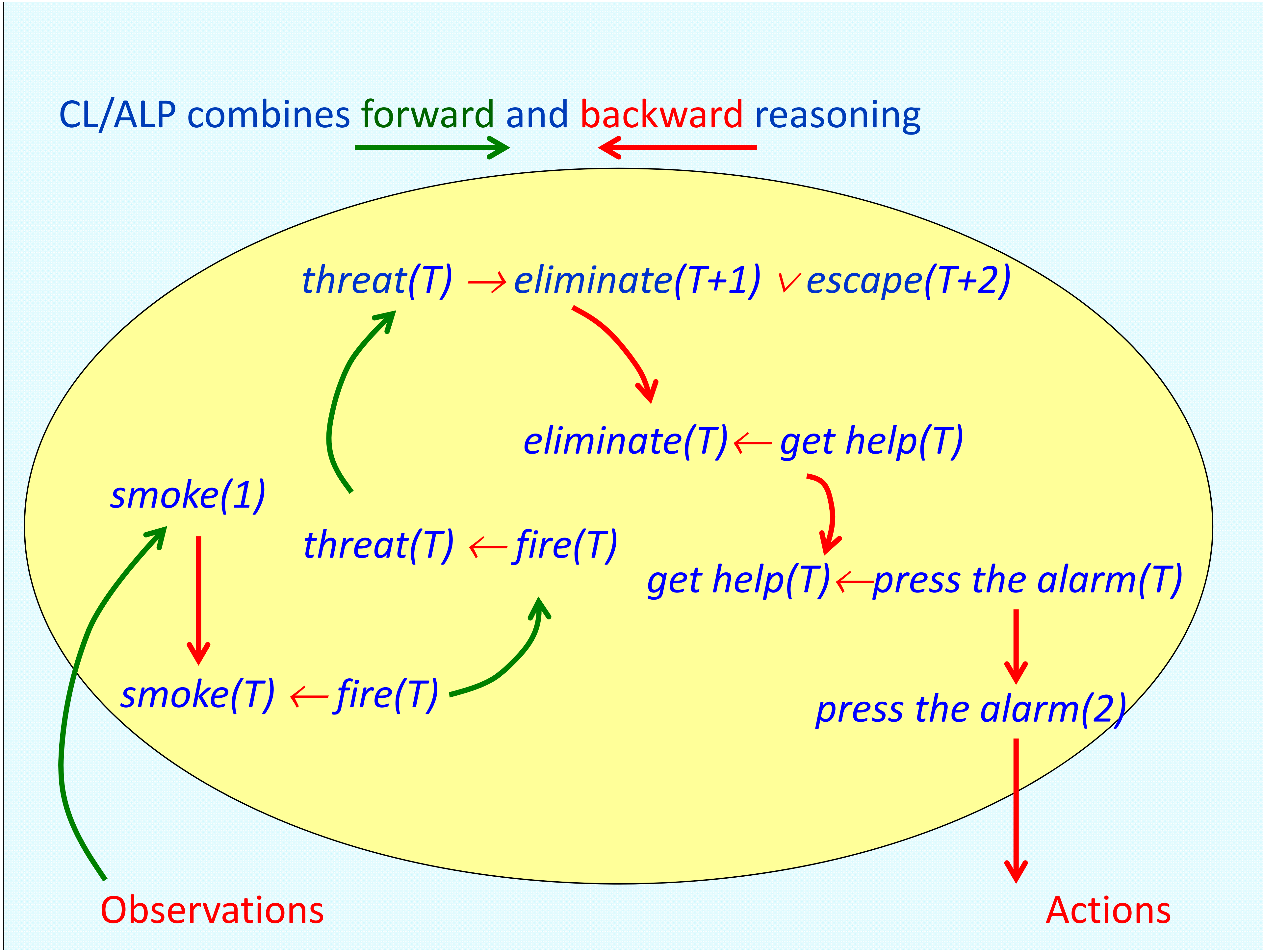
# Outline of the talk

Overview

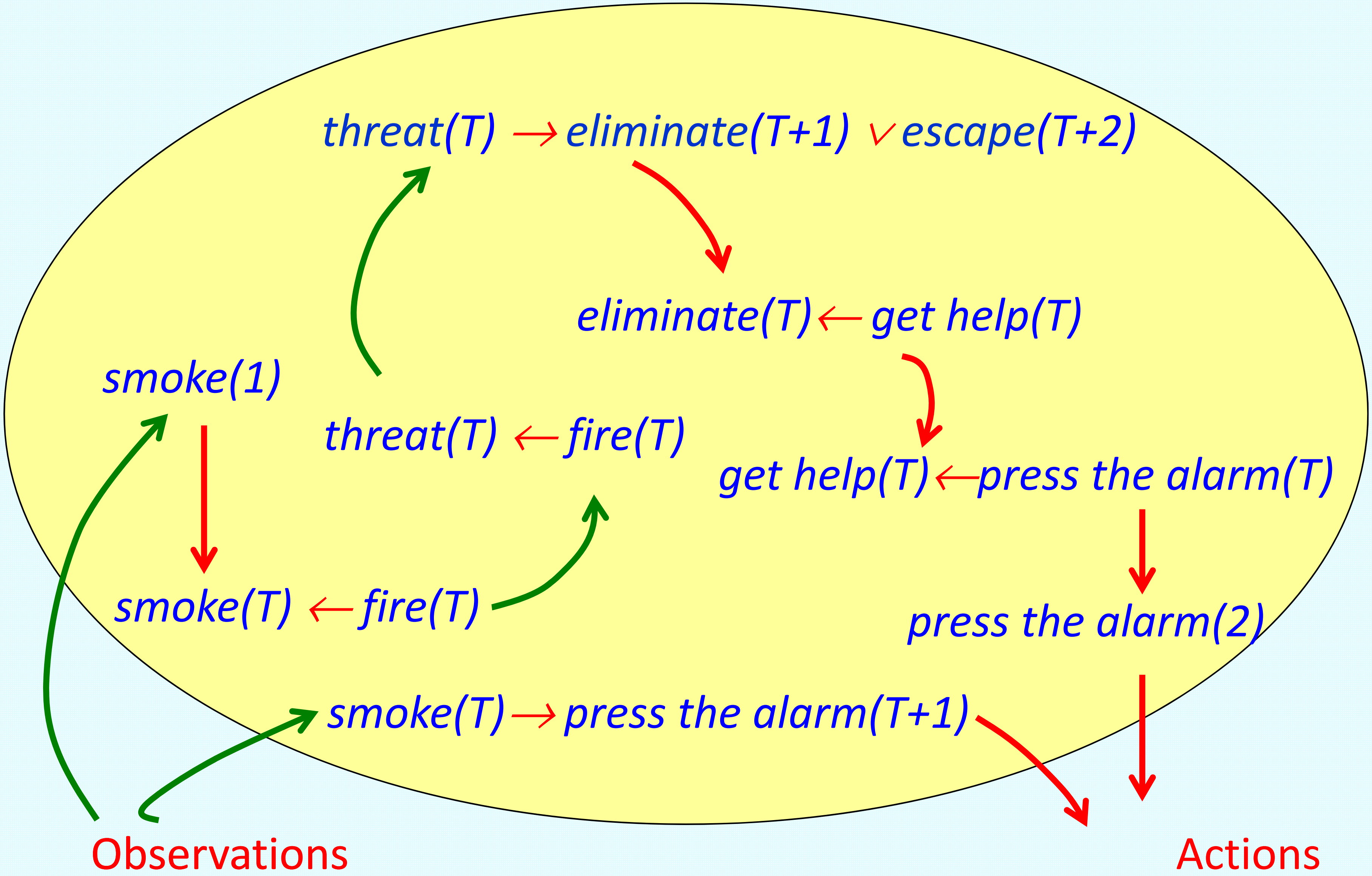
Logic programs represent beliefs

Production systems represent goals (but have no logic)

Computational Logic combines goals and beliefs
embedded in an observe-think-decide-act agent cycle


CL/ALP combines forward and backward reasoning
threat(T) → eliminate(T+1) ∨ escape(T+2)
eliminate(T) ← get help(T)
smoke(1)
threat(T) ← fire(T)
get help(T) ← press the alarm(T)
smoke(T) ← fire(T)
press the alarm(2)
Observations
Actions

CL/ALP is compatible with the dual process theory



# Abductive Logic Programming (ALP)

Goal G: $threat(T) \rightarrow eliminate(T+1) \vee escape(T+2)$

Beliefs B: $threat(T) \leftarrow fire(T)$

$smoke(T) \leftarrow fire(T)$

$eliminate(T) \leftarrow get\ help(T)$

$get\ help(T) \leftarrow press\ the\ alarm(T)$

Observation O: $smoke(1)$

# Abductive Logic Programming (ALP)

| | |
|---|---|
| Goal G: | *threat(T)* $\rightarrow$ *eliminate(T+1)* $\vee$ *escape(T+2)* |
| Beliefs B: | *threat(T)* $\leftarrow$ *fire(T)*<br>*smoke(T)* $\leftarrow$ *fire(T)*<br>*eliminate(T)* $\leftarrow$ *get help(T)*<br>*get help(T)* $\leftarrow$ *press the alarm(T)* |
| Observation O: | *smoke(1)* |
| Assumptions $\Delta$: | *fire(1)*<br>*press the alarm(2)* |
| Abduction: | *fire(1)* explains *O* |
| Planning: | *press the alarm(2)* achieves *G* |

# Abductive Logic Programming (ALP)

Goal G: *threat(T)* $\rightarrow$ *eliminate(T+1)* $\vee$ *escape(T+2)*

Beliefs B: *threat(T)* $\leftarrow$ *fire(T)*
*smoke(T)* $\leftarrow$ *fire(T)*
*eliminate(T)* $\leftarrow$ *get help(T)*
*get help(T)* $\leftarrow$ *press the alarm(T)*

Observation O: *smoke(1)*

Assumptions $\Delta$: *fire(1)*
*press the alarm(2)*

Abduction: *fire(1)* explains *O*

Planning: *press the alarm(2)* achieves *G*

G $\cup$ O is true in the model of the world determined by B $\cup$ $\Delta$.

# ALP - Different $\Delta$ can solve the same task.

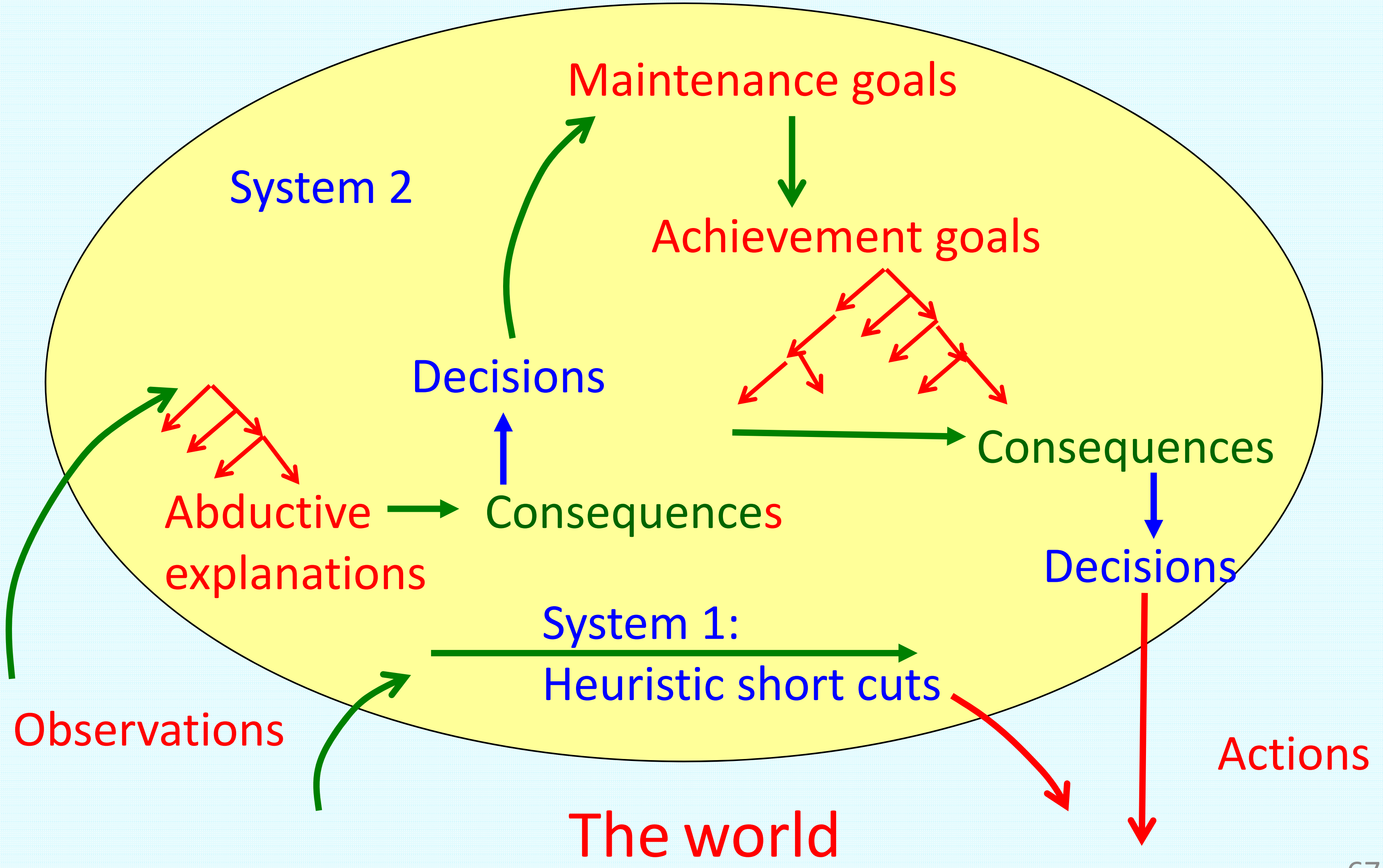
The challenge is to find the best $\Delta$
within the computational resources available.

In classical decision theory, actions are evaluated
by the expected utility of their consequences.

In philosophy of science, explanations are evaluated
by their probability and explanatory power.
(The more observations explained the better.)

In ALP, actions and assumptions are combined in $\Delta$,
and are treated in the same way, and
forward reasoning is used to derive their possible consequences,

# Computational Logic as a unifying framework



# Conclusions

## Computational Logic

- combines goal and beliefs
- inspired by models of human thinking and decision making
- provides a foundation for more human-oriented computing
- can help people think and communicate more effectively.

- CL as Generator of Human Action
- CL as Language of Thought (LOT)
- CL as a unifying framework

# How to obtain evidence about the Nature of Human Thought?

Study natural language texts designed to be easy to understand.

The London Underground Emergency Notice

Study advice about effective natural language communication.

The Pyramid Principle
Joseph Williams: Toward Clarity and Grace

## How to obtain evidence about the Language of Thought (LOT)?

Study natural language texts designed to be easy to understand.

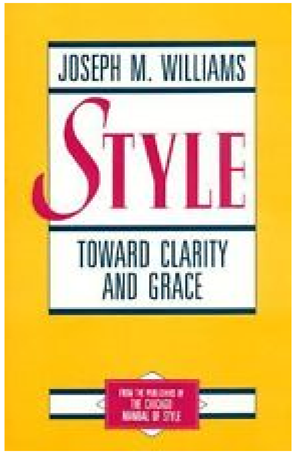
The London Underground Emergency Notice

Study advice about effective natural language communication.

The Pyramid Principle
Joseph Williams: Toward Clarity and Grace

Not: The teacher gave the student a good mark.
She was happy.

But: The teacher was happy.
Or: The student was happy.




Not: Our lack of knowledge of the topic of the talk prevented us from understanding it.

Or: Because we did not know the topic of the talk , we could not understand the talk.

I.E. A person cannot understand a talk
if the person does not know the topic of the talk.
We did not know the topic of the talk.

# Williams: Two Principles of Coherence

1. Put at the beginning of a sentence those ideas that you have already mentioned, referred to, or implied, or concepts that you can reasonable assume your reader is already familiar with, and will readily recognise.

2. Put at the end of your sentence the newest, the most surprising, the most significant information: information that you want to stress – perhaps the information that you will expand on in your next sentence.

# Coherence

Example: *A.*
*If A then B.*
*If B then C.*
Therefore *C.*

Example: *C?*
*C if B.*
*B if A.*
*A.*
Therefore *C.*

# In CL, goals and beliefs are combined in a connectionist network

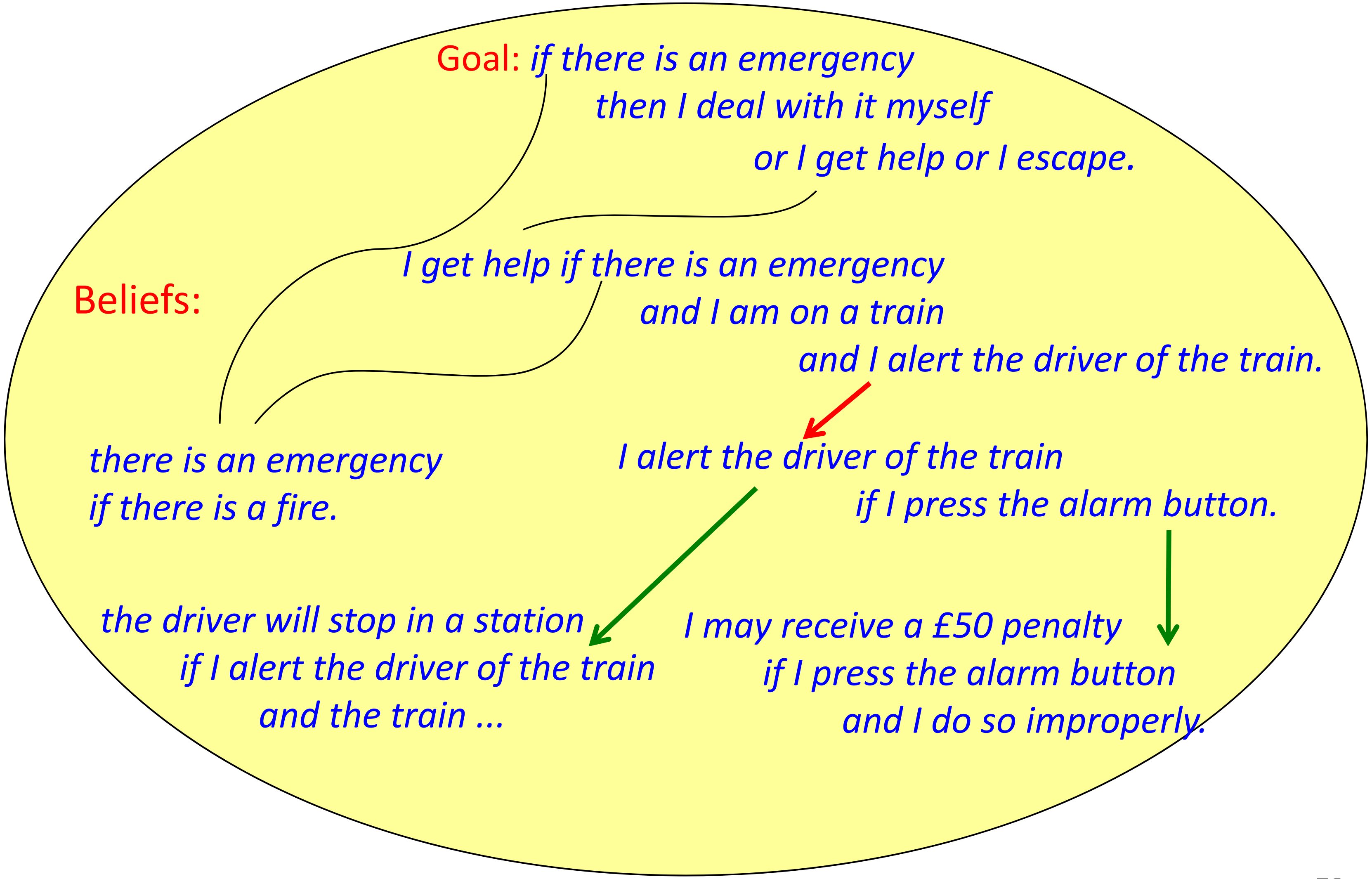
Goal: *if there is an emergency*
*then I deal with it myself*
*or I get help or I escape.*

Beliefs:

*I get help if there is an emergency*
*and I am on a train*
*and I alert the driver of the train.*

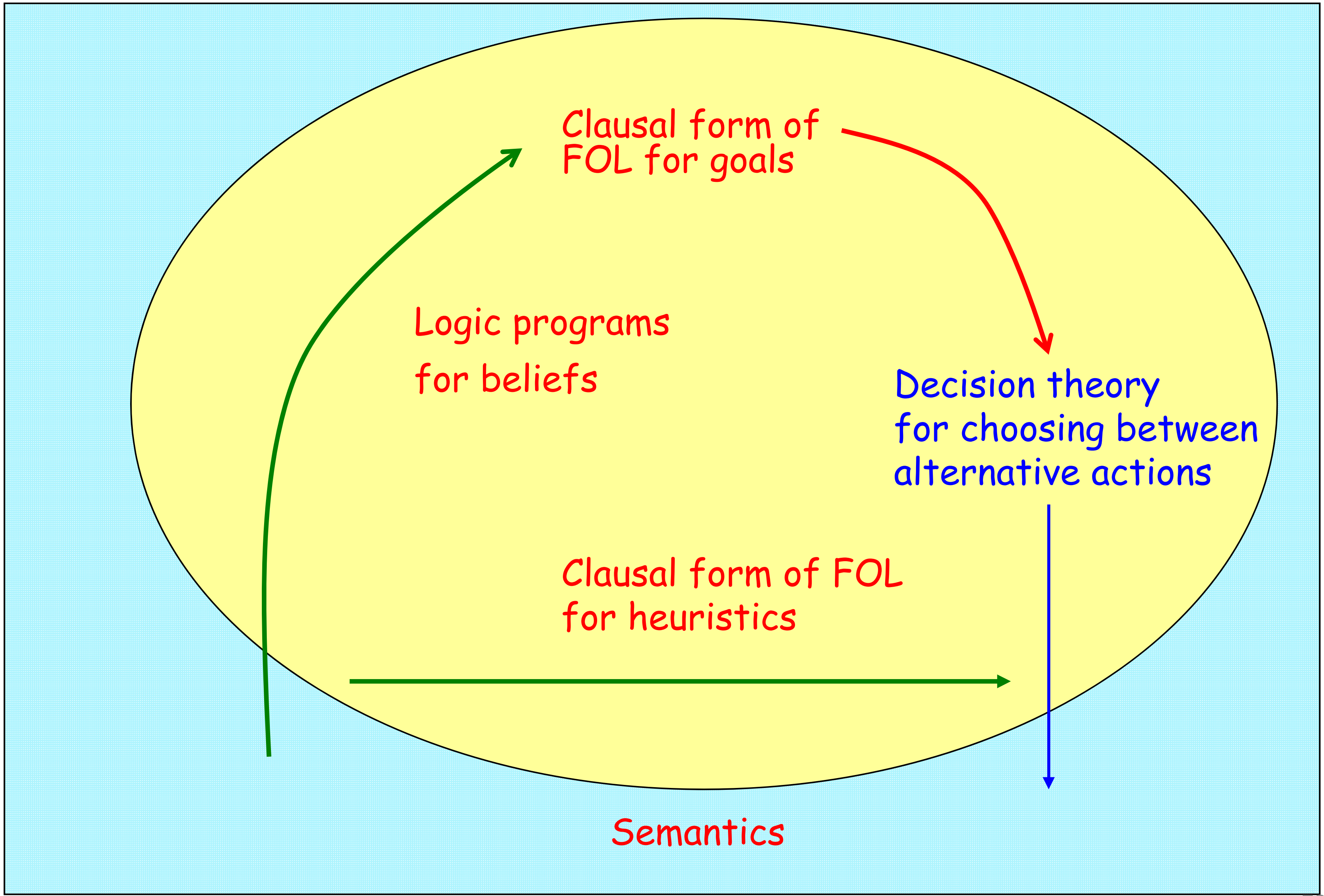
*there is an emergency*
*if there is a fire.*

# To express yourself coherently, connect new ideas with existing ideas.



- CL as Generator of Human Action
- CL as Language of Thought (LOT)
- CL as a unifying framework

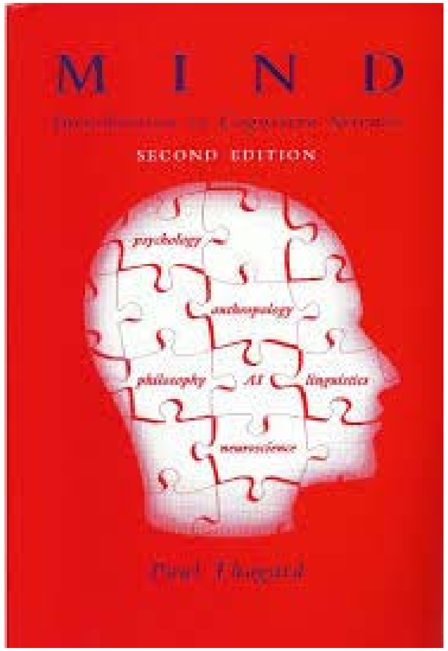
# The CL Agent Model as a unifying framework



# “Rule-based systems” (production systems) as an alternative model of human thinking

“Unlike logic, rule-based systems
can easily represent
strategic information
about what to do”:

*If you want to go home*
*and you have the bus fare,*
*then you can catch a bus.*




But this misses the real logic of the strategy:

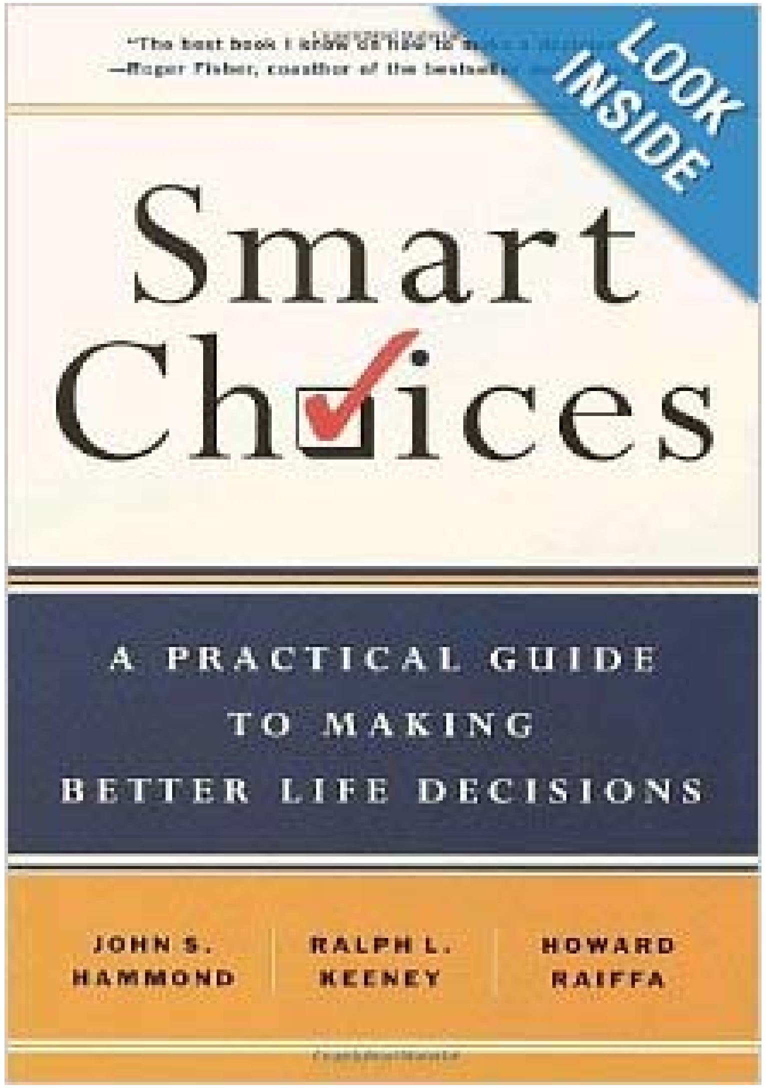
*You go home if you have the bus fare and you catch a bus.*

Backward reasoning with this logic behaves like
forward reasoning with the rule.

**Smart Choices:**

73 customer reviews

5 star – 49
4 star – 17
3 star – 3
1 star - 1



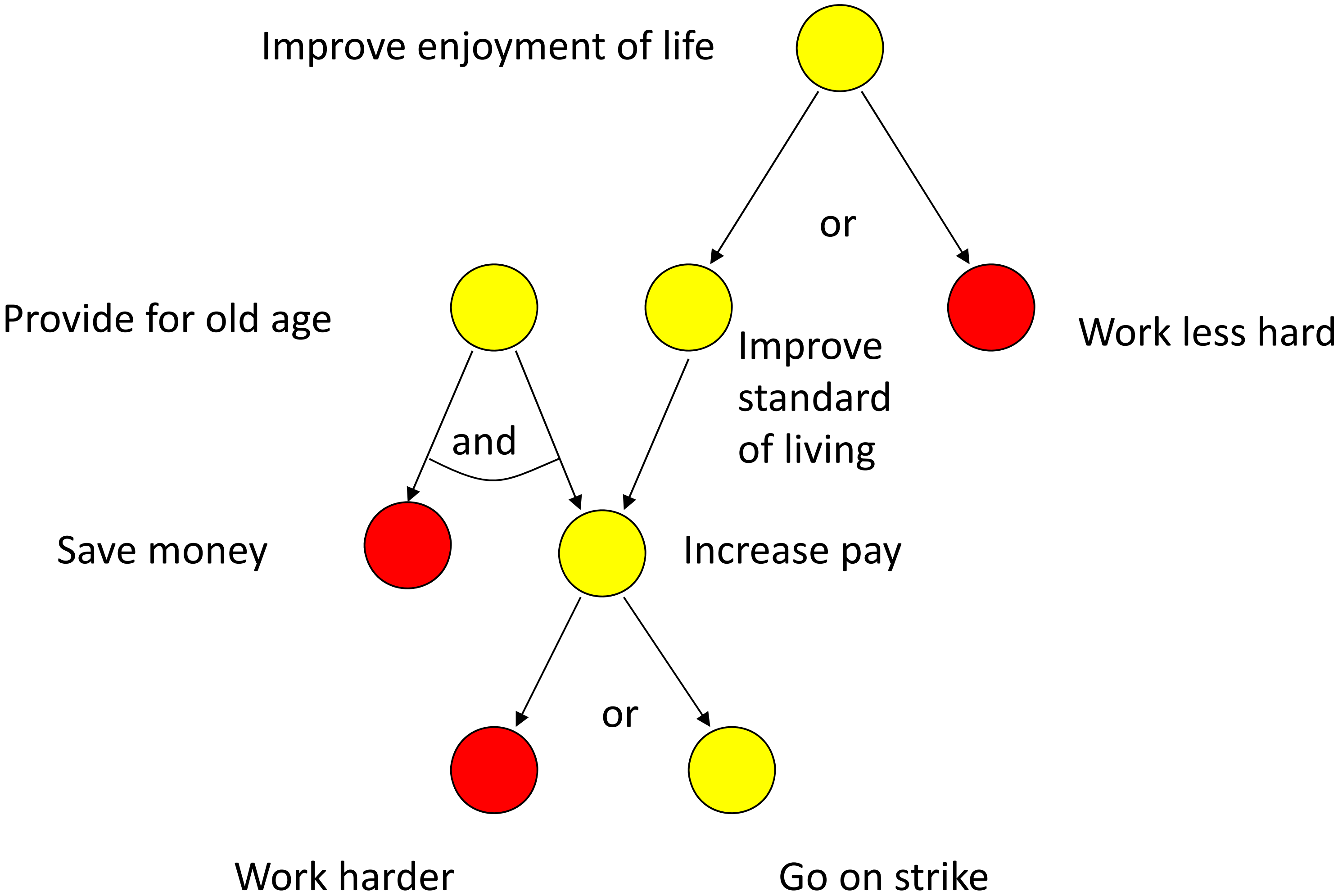
# Smart choices – a better decision theory

Classical decision theory assumes that all of the alternative actions are fixed and given in advance.

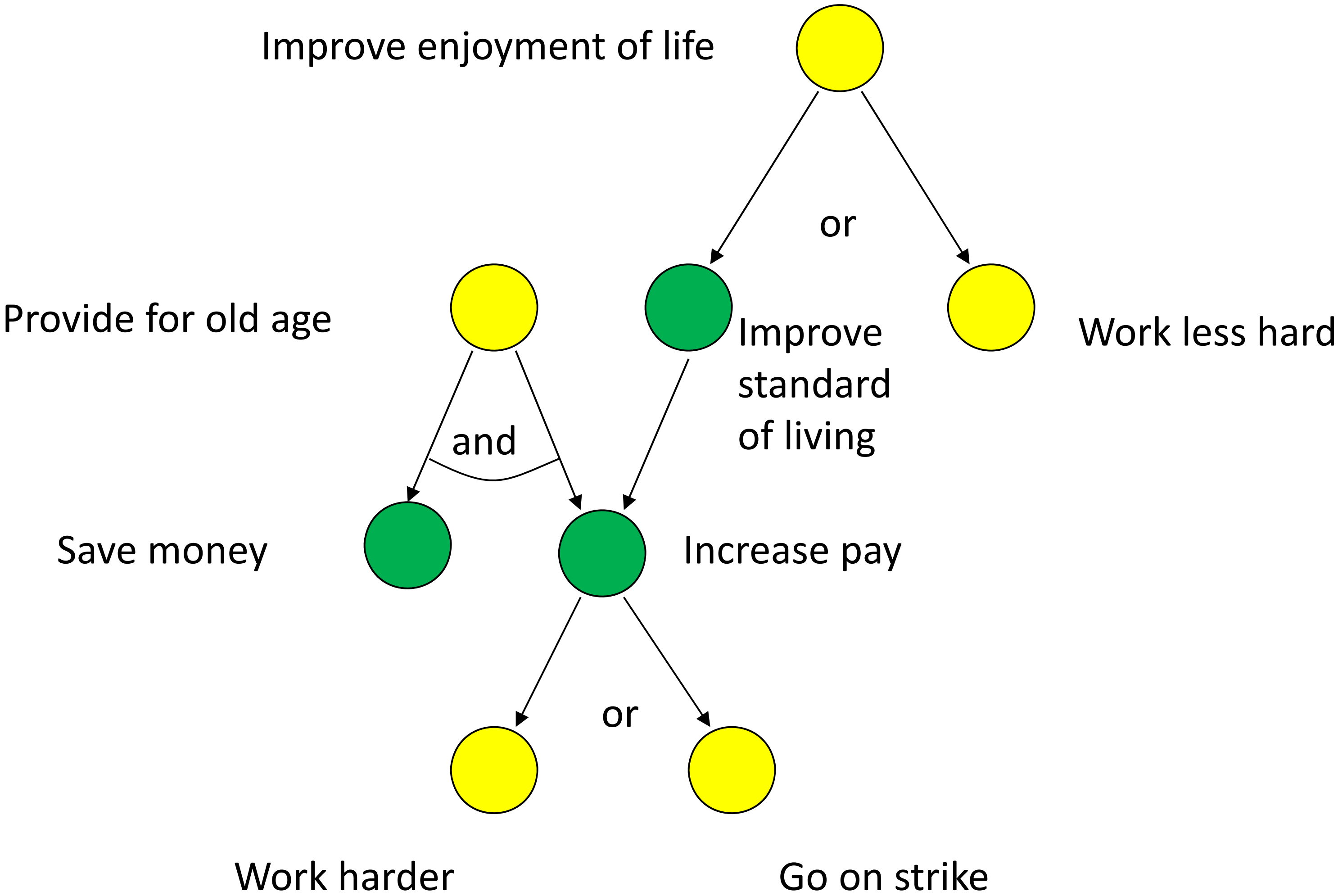
To make smarter decisions:

- identify the goals that motivate the alternatives
- identify the beliefs that reduced the goals to actions
- judge whether the beliefs are true
- investigate whether there are any other relevant true beliefs
- investigate whether there are any other relevant goals
- identify events that can trigger motivating goals
  and prepare for them before they happen.

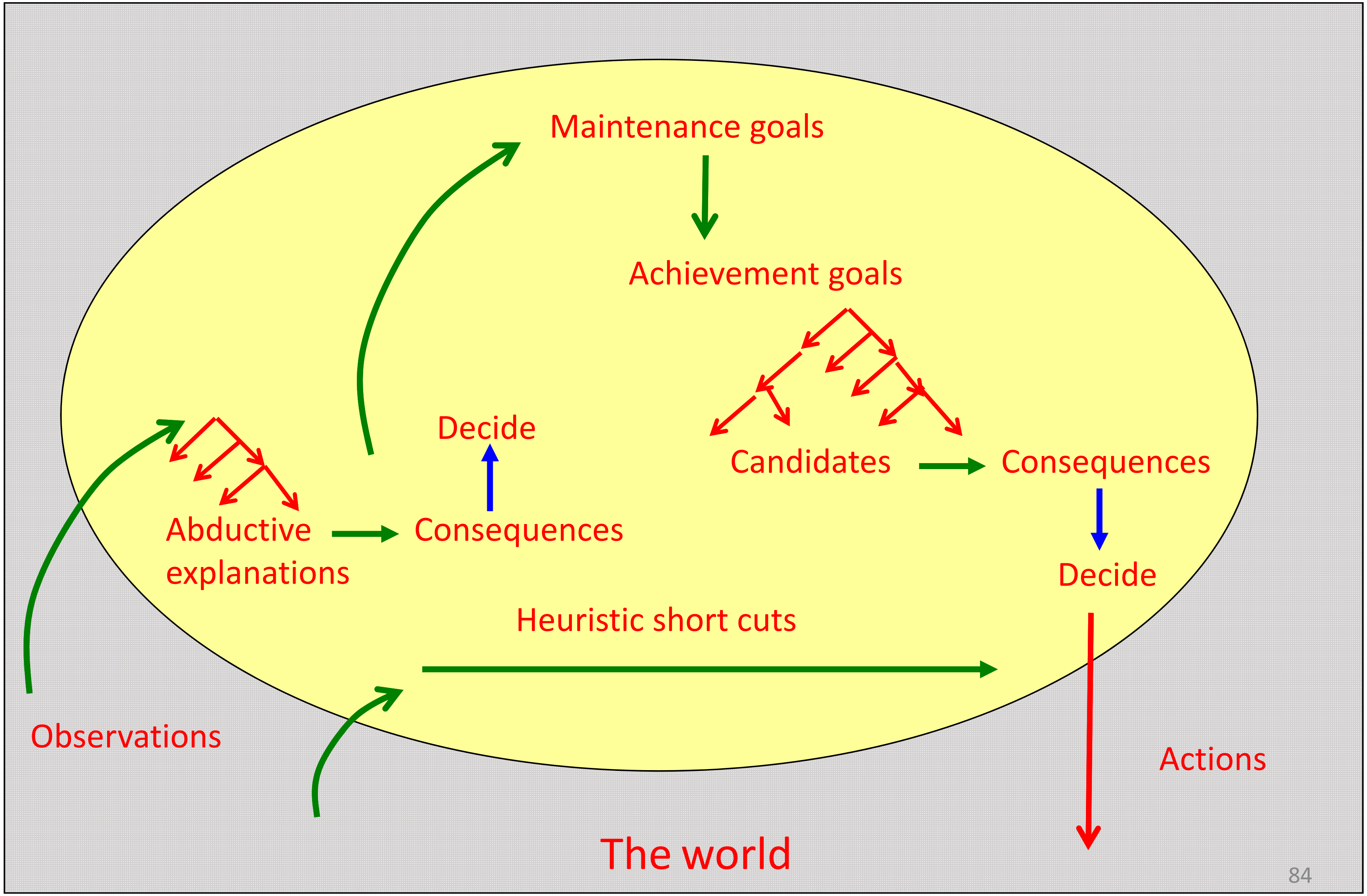
## Conflicting ways of solving different goals can sometimes be resolved by finding alternative solutions



## Conflicting ways of solving different goals can sometimes be resolved by finding alternative solutions



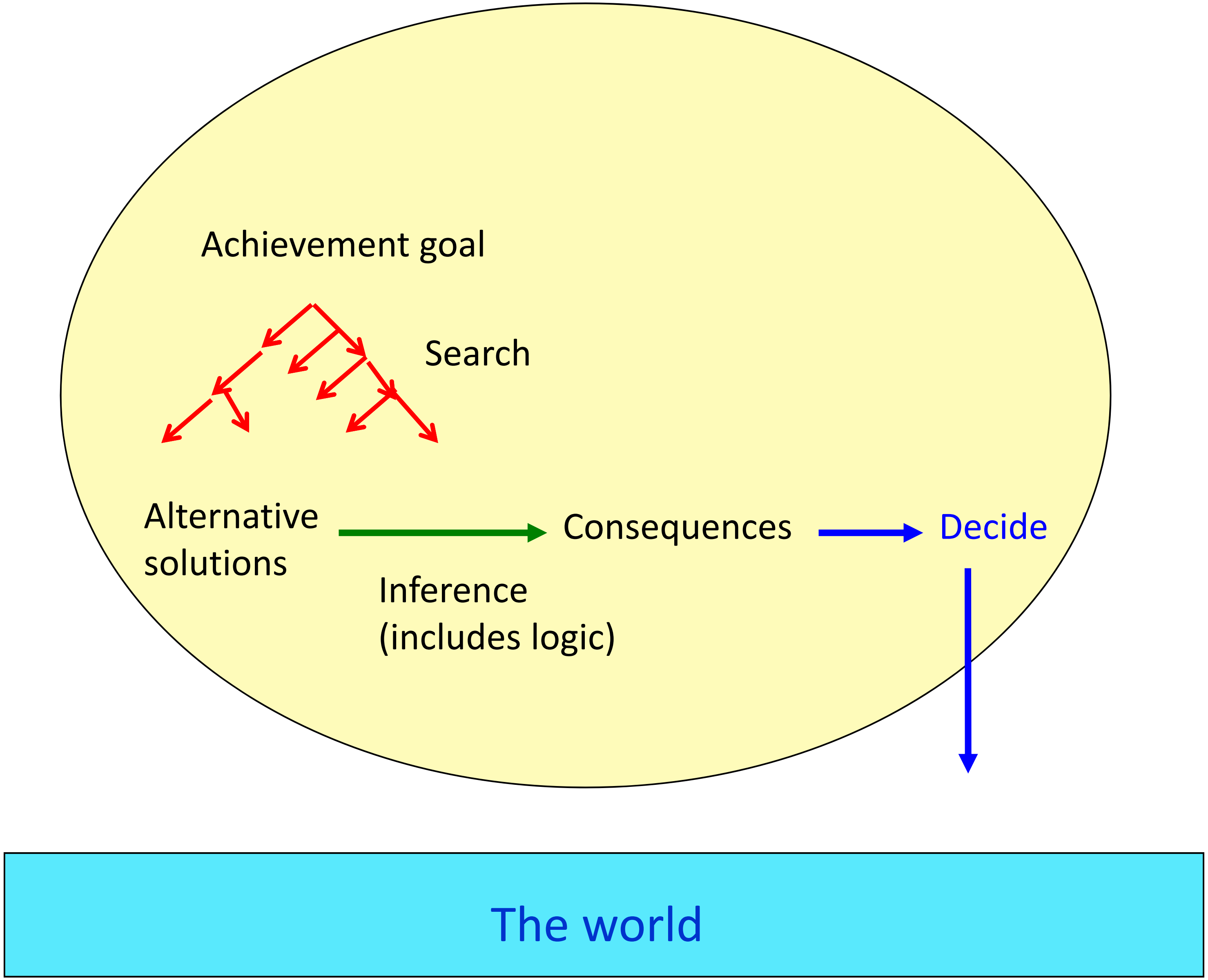
# Conclusion: Computational Logic as a unifying framework



# Conclusions

- The Computational Logic combines and unifies
  - Logic
  - Connectionism
  - Production Systems
  - Decision Theory
- Computational Logic can help people
  - communicate better
  - make smarter decisions
- Computational Logic can help computer scientists and engineers
  - develop more human-oriented computer languages
  - more intelligent computer applications

# Baron's view of search in relation to thinking and deciding



Conflicting ways of solving different goals can sometimes be resolved by finding alternative solutions e.g.

Achievement goals:

Improve enjoyment of life

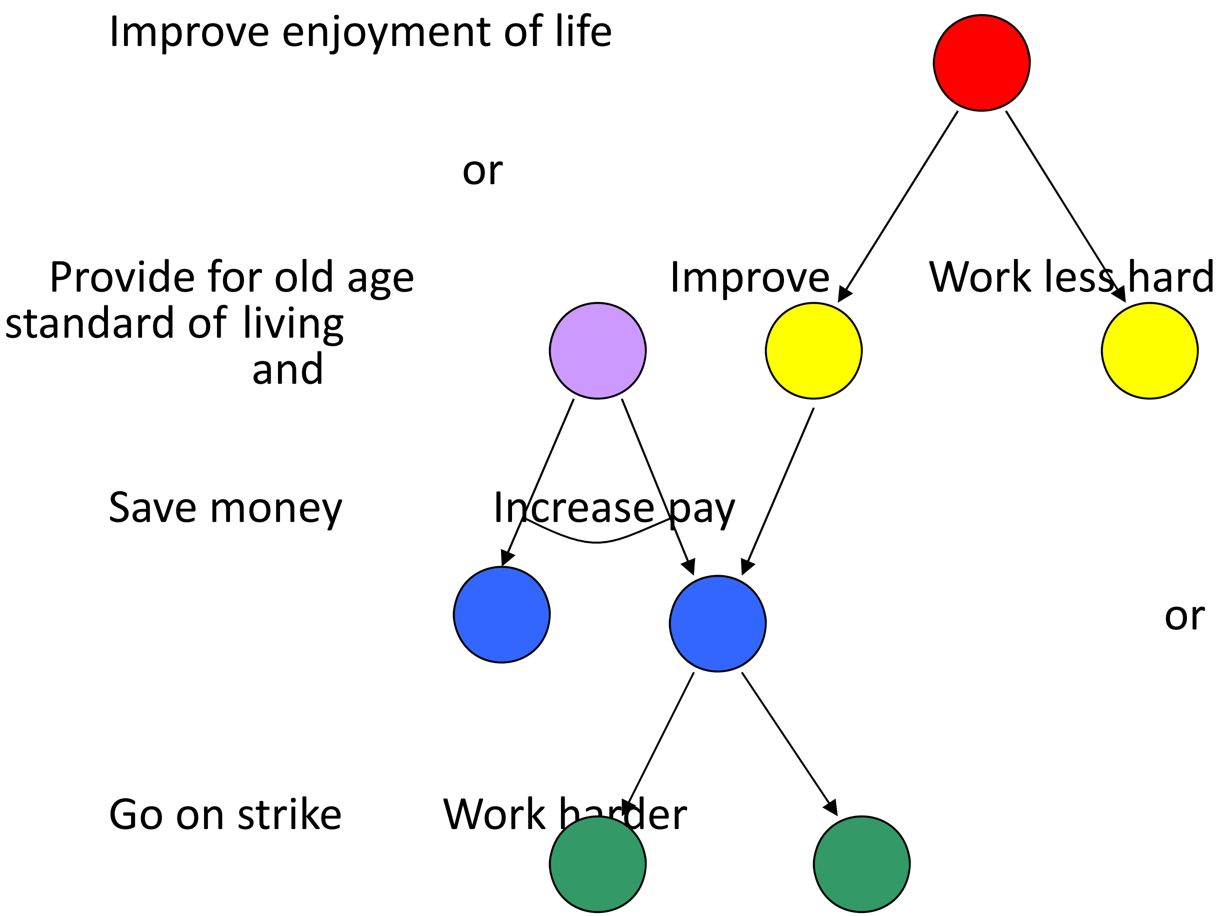
Provide for old age

Beliefs:

You improve enjoyment of life
if you work less hard.

You provide for old age,
If you save money and work harder.

## Conflicting ways of solving different goals can sometimes be resolved by finding alternative solutions



# smart choices

given alternative choices,
analise goals affected by the choices
and other ways of solving the goals
choose a solution(s) that maximises all the goals,
possibly generating an alternative to the original choice.

## Complex decisions can often be replaced by heuristic rules

Instead of the high-level maintenance goals:

*If a person attacks me,*
*then I attack the person or I get help or I try to escape.*

and complex decision between the actions:

*I attack the person* or
*I get help* or
*I try to escape*

we can employ simpler, lower-level heuristic maintenance goals in logical form:

*If a person attacks me and I am stronger than the person,*
*then I attack the person*

*If a person attacks me and I am weaker than the person,*
*then I get help*

*If a person attacks me and I and my helpers are weaker than the person,*
*then I try to escape*

## The network of goals and beliefs can use information about previously useful connections

- Links can have forward or backward directions.

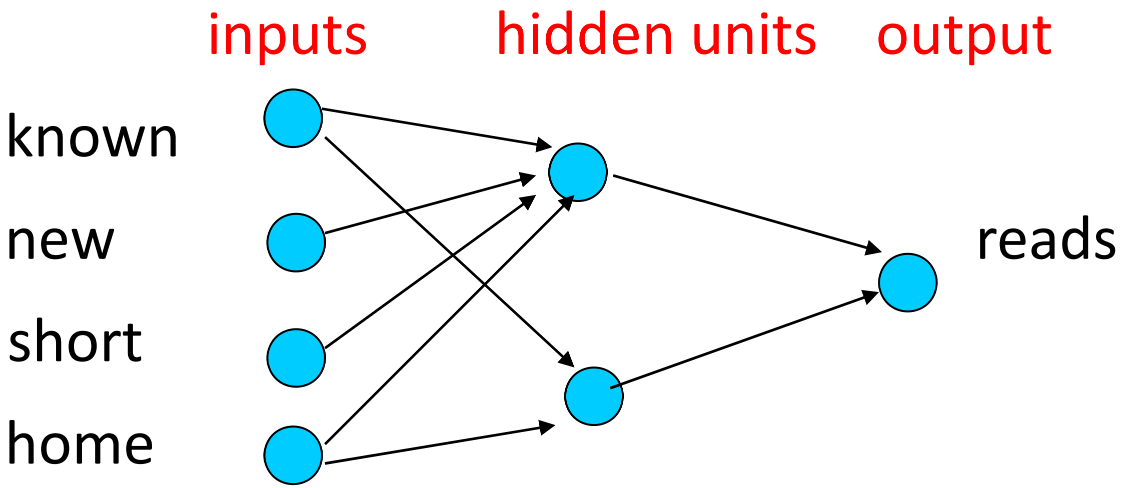
- Links can be weighted by statistics about how often they have been used successfully in the past.

- Input observations and goals can be assigned different strengths (or utilities).

- The strength of observations and goals can be propagated through the graph in proportion to the weights on the links.

- Activating links with the highest weighted strengths is like the activation networks of Patie Maes.

Feed-forward neural networks can be represented as logic programs (from Computational Intelligence, Poole, Mackworth, Goebel, 1998)




*reads with strength W*

*if arguably reads with strength W1*

*and arguably doesn't read with strength W2*

*and* $W = f(2.98 + 6.88W1 - 2.1W2)$

*arguably reads with strength W1*
*if known with strength W4*
*and new with strength W5*
*and short with strength W6*
*and home with strength W7*
*and* $W1 = f(-5.25 + 1.98W4 + 1.86W5 + 4.71W6 - .389W7)$

*arguably doesn't read with strength W2*
*if known with strength W4*
*and new with strength W5*
*and short with strength W6*
*and home with strength W7*
*and* $W2 = f(.493 - 1.03W4 - 1.06W5 - .749W6 + .126W7)$

## In English

*A person will read a paper*
*if there is strong reason to read the paper and*
*there is no sufficiently strong reason not to read the paper.*

*There is a reason to read the paper*
*if the author is known to the person, the topic is new,*
*the paper is short and the person is at home.*

*There is a reason not to read the paper*
*if the author is not known to the person, the topic is old,*
*the paper is long and the person is not at home.*

Discovery. Sometimes writers put their main POINT sentences . last because they want theIr readers to work through an argument or a body of data to experience a sense of discovery. They believe that the development of the POINT is as important as the POINT itself. In fact, that kind of organization characterizes parts of this book: we have frequently begun with some contrasting passages to develop a small-p point, in the hope that you would grasp it a moment before you read the POINT sentence.

As we have emphasized, though" most readers in most professional contexts prefer documents with main POINT early. Articles in many sciences hard or soft begin with abstracts that typi- - cally contain the POINT of the article. Readers in those areas also know that, after reading the abstract, they can go directly to the conclusion if they want to see the main POINT expressed in more detail. These readers employ a reading strategy that creates a POINT-first form: if they don't find the POINT on the first page, they flip to the conclusion, where they expect to find it.

# Clausal logic is a simplified form of first-order logic (FOL)

In clausal logic, sentences have a simplified form, e.g.:

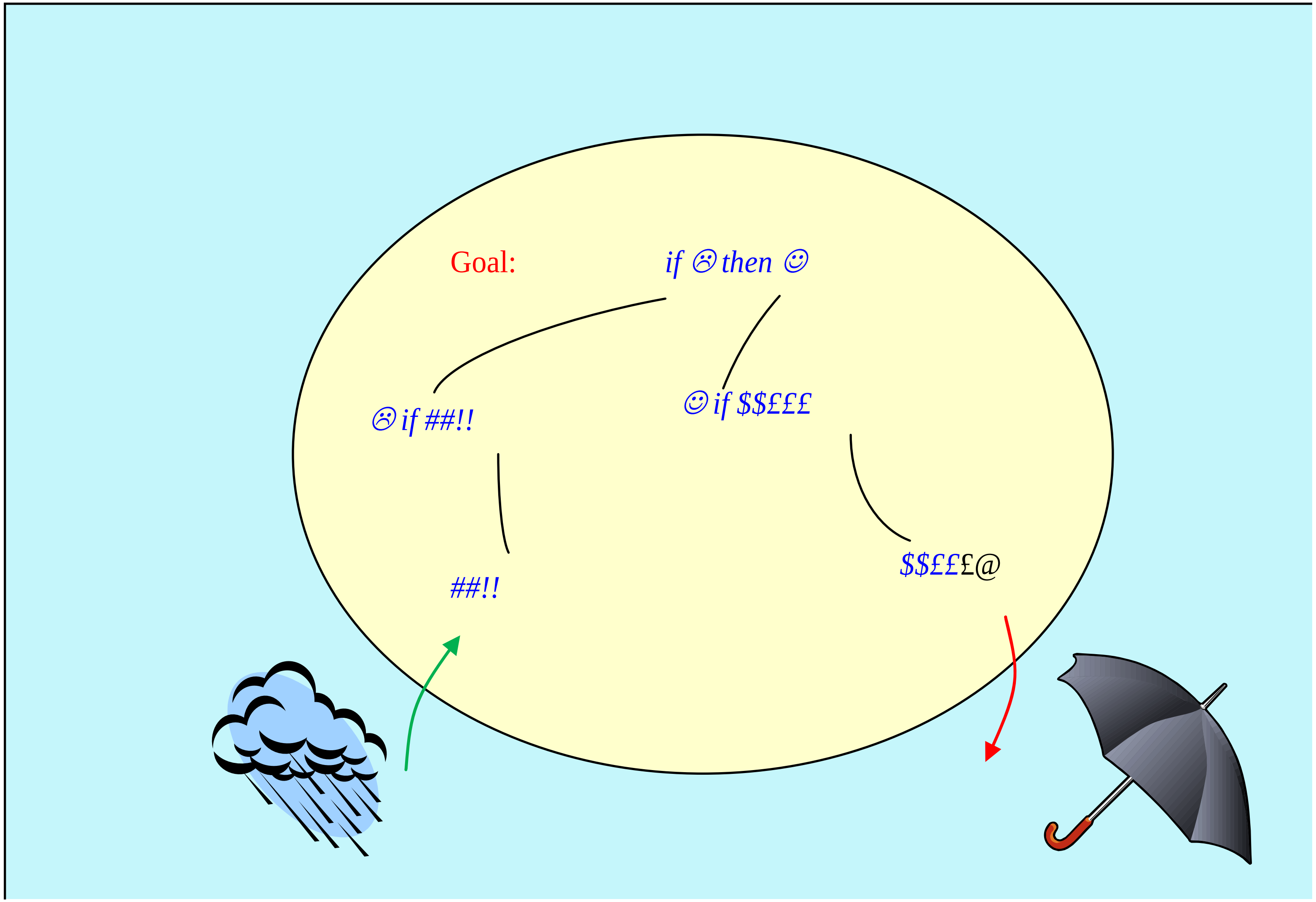
*has-feathers(X) ← bird(X).*
*bird(john).*

In standard FOL, the same beliefs can be expressed in infinitely many, equivalent ways, including:

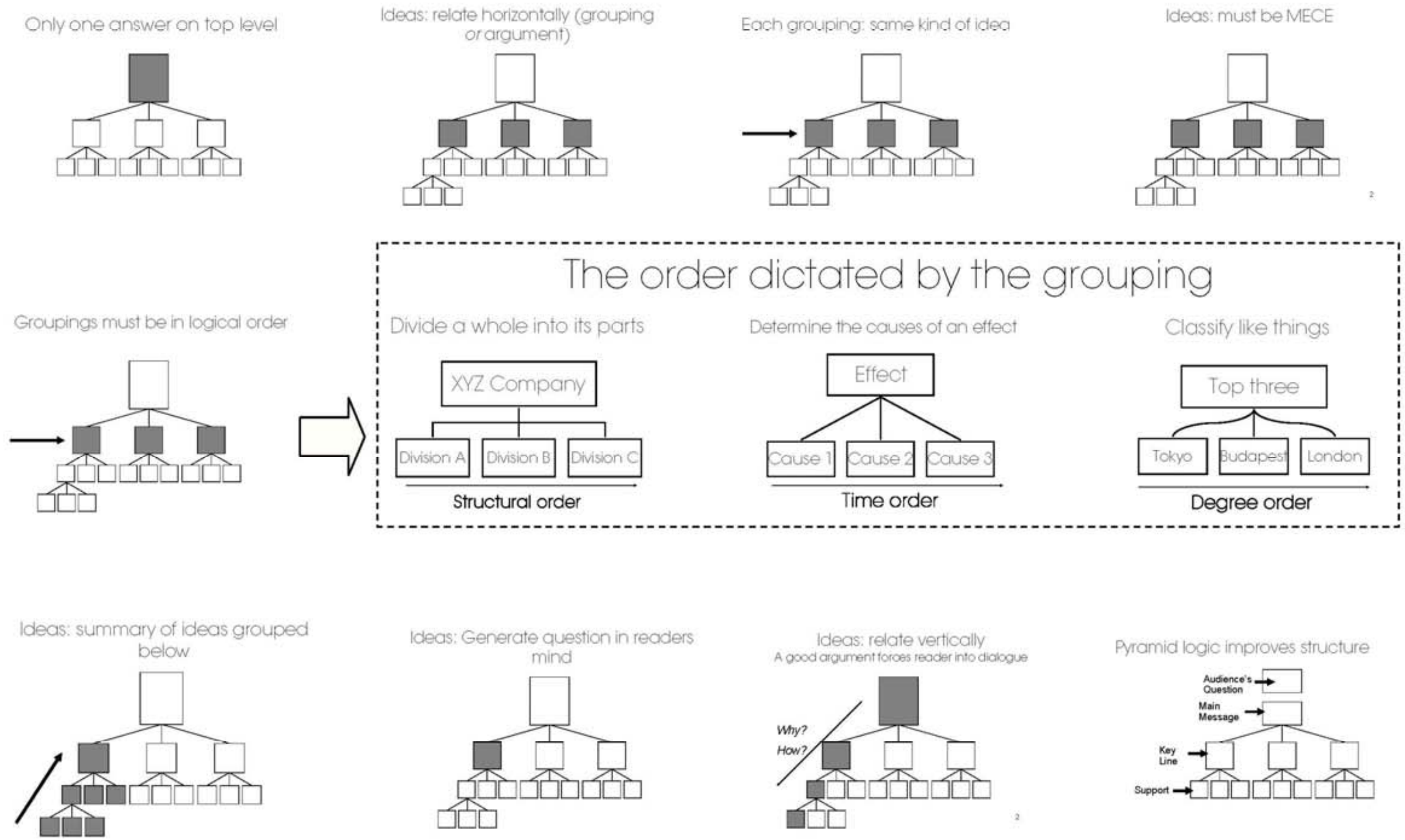
$\neg(\exists X((\neg has\text{-}feathers\ (X) \wedge bird(X)) \vee \neg bird(john)))$
$\neg(\exists X((\neg has\text{-}feathers\ (X) \vee \neg bird(john)) \wedge\ (bird(X) \vee \neg bird(john))))$

In clausal logic, reasoning is simpler than in standard FOL and can be reduced to forward or backward reasoning.

# It can be difficult or impossible to put thoughts into words



# Ideas in writing should always form a pyramid




jazzcode.org

## Clausal logic as a theory of the LOT can help people to communicate more effectively

By expressing communications:

| | |
|---|---|
| Clearly | So that their meaning is unambiguous. |
| Simply | So that their meaning is close to their canonical form. |
| Coherently | So that it is easy to link new information to old information. |

# The syntax of logic programs

*Clauses* have the form:

$$\text{conclusion} \leftarrow \textit{condition}_1 \wedge \textit{condition}_2 \; .... \wedge \textit{condition}_n$$

or $\forall X \, [\, \textit{condition}_1 \wedge \textit{condition}_2 \; .... \wedge \textit{condition}_n \rightarrow \textit{conclusion} \,]$

i.e. *for all X, conclusion if condition*$_1$ *and condition*$_2$ .... *and condition*$_n$

where *conclusion* is an atomic formula

and *condition*$_i$ are atomic formulas or negations of atomic formulas.

If $n = 0$, then the clause is a "fact"

*conclusion if true*

i.e. *conclusion*

If *conclusion* and all *condition*$_i$ are atomic formulas,
then the clause is a Horn clause.

# The syntax of maintenance goals = a variant of the clausal form of first order logic

Goals: clauses of the form:

$\forall X$ [$condition_1 \wedge condition_2$ .... $\wedge$ $condition_n \rightarrow$
$\exists Y$ [$conclusion_1 \vee conclusion_2$ .... $\vee conclusion_m$]]

where $X$ is the set of all variables that occur in the $condition_i$
and $Y$ is the set of all variables that occur only in the $conclusion_j$

If $m = 0$, then the goal is equivalent to a *denial* (or *constraint*):

$$condition_1 \wedge condition_2 \; .... \wedge condition_n \rightarrow false$$

i.e. $\neg$ [$condition_1 \wedge condition_2$ .... $\wedge$ $condition_n$ ]

It can sometimes be hard to tell the difference between a goal and a belief.

# British Nationality Act 1981

## 1981 CHAPTER 61

An Act to make fresh provision about citizenship and nationality, and to amend the Immigration Act 1971 as regards the right of abode in the United Kingdom.
[30th October 1981]

BE IT ENACTED by the Queen's most Excellent Majesty, by and with the advice and consent of the Lords Spiritual and Temporal, and Commons, in this present Parliament assembled, and by the authority of the same, as follows:—

### PART I

### BRITISH CITIZENSHIP

*Acquisition after commencement*

**1.**—(1) A person born in the United Kingdom after commencement shall be a British citizen if at the time of the birth his father or mother is— Acquisition by birth or adoption.

(*a*) a British citizen; or

(*b*) settled in the United Kingdom.

(2) A new-born infant who, after commencement, is found abandoned in the United Kingdom shall, unless the contrary is shown, be deemed for the purposes of subsection (1)—

(*a*) to have been born in the United Kingdom after commencement; and

(*b*) to have been born to a parent who at the time of the birth was a British citizen or settled in the United Kingdom.

1.-(1) A person born in the United Kingdom after commencement shall be a British citizen if at the time of the birth his father or mother is –

(a) a British citizen; or

(b) settled in the United Kingdom.

The meaning of subsection 1.-(1)

*A person shall be a British citizen by 1.-(1)*

| | |
|---|---|
| *if* | *the person was born in the United Kingdom* |
| *and* | *the person was born after commencement* |
| *and* | *a parent of the person was a British citizen at the time of the person's birth or* |
| | *a parent of the person was settled in the United Kingdom at the time of the person's birth.* |

## Heuristics are often represented as *condition-action* rules in production systems

Declarative "working memory" consisting of atomic sentences, and

Procedures consisting of condition-action rules:

*If conditions C, then do actions A.*

Procedures look like logical conditionals,

but do not have a logical semantics.

Production system cycle:

- observe a current input
- use *forward chaining* to match the input with a condition in *C*
- use *backward chaining* to verify the remaining conditions of *C*
- perform conflict-resolution to choose a single rule if the conditions *C* of more than one rule are satisfied, and
- execute the associated actions *A*.

# Maintenance goals generate actions

*If a person attacks me,*
*then I fight back or I get help or I try to escape.*

Given an observation or consequence of an observation:

*john attacks me*

Reason forwards to derive the achievement goal:

*I fight back or I get help or I try to escape*

Decide between the different actions:

*I attack the person* or *I get help* or *I try to escape*

# How are thinking and logic related?

In the philosophy of language, there are three main theories:

Human thinking does not have a language-like structure at all.
So communicating thoughts from writer to reader is almost a miracle.

The LOT is a form of the public, natural language that we speak.
So communicating thoughts from writer to reader is trivial.
Just say what you think.

The LOT is a private language-like representation,
which does not depend on the natural language that we speak.
So communications can be improved by expressing them in a form
that is close to the language of thought, because this will reduce the amount
of effort the reader needs to translate communications into thoughts.

I will argue that the LOT is a private, language-like representation that has a simplified logical form, which has a connectionist structure.

# How to investigate the LOT? Part 1 of 2

According to relevance theory [Sperber and Wilson, 1986], people understand natural language by attempting to extract the most information for the least effort.

It follows that:
If you want to find out whether there is a LOT, and what it is like, then study natural language texts that communicate useful information and are easy to understand.

Understanding the LOT can help us:

- communicate more effectively with other people
- develop better computer languages

# How to investigate the LOT? Part 2 of 2

According to relevance theory,
people understand natural language by attempting
to extract the most information for the least effort.

It follows that:

If you want to find out whether there is a LOT, and what it is like,
then study advice about effective natural language communication.

Understanding the LOT can help us:

- to communicate more effectively with other people
- to develop better computer languages

# To express yourself effectively in natural language

1. Avoid ambiguity. e.g.

   Not: The teacher gave the student a good mark.
   She was happy.

   Better: The teacher was happy with the student's work.
   Or: The student was happy with the good mark.

   } clarity

2. Avoid unnecessary complexity. e.g.

   Not: Our lack of knowledge of the topic of the talk
   prevented us from understanding it.

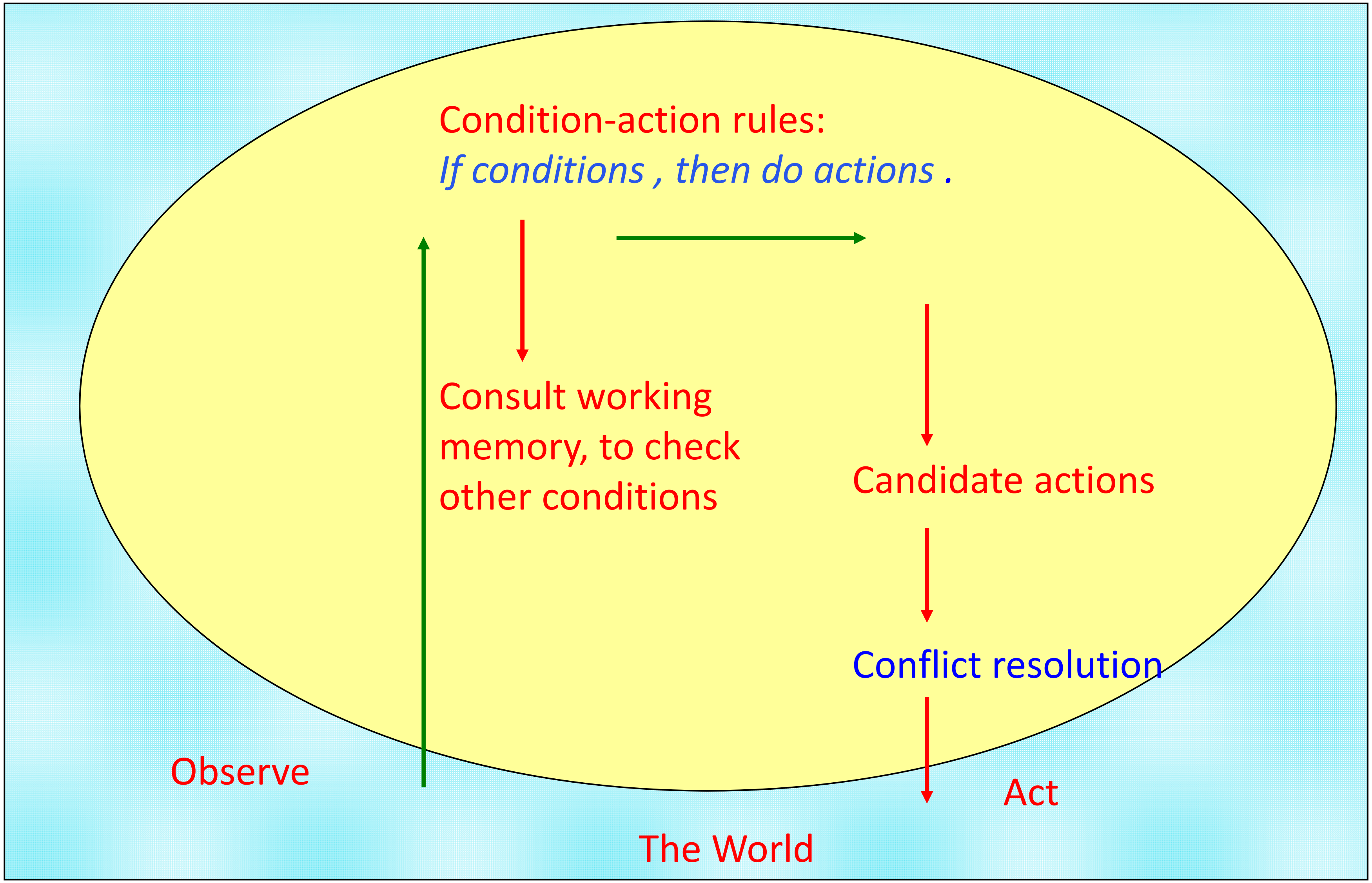
   Better: Because we did not know the topic of the talk ,
   we could not understand the talk.

   Or: A person cannot understand a talk
   if the person does not know the topic of the talk.
   We did not know the topic of the talk.

   } simplicity

3. Connect related ideas together. } coherence

# The production system cycle



# Conflict resolution

Several conflicting actions can be derived at the same time.

For example:

*If someone attacks me, then attack them back.*
*If someone attacks me, then get help.*
*If someone attacks me, then try to escape.*

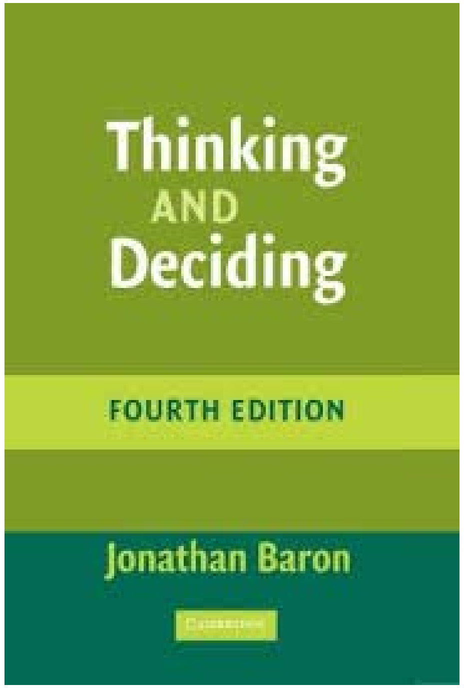
The agent needs to use "conflict resolution" to *decide* what to do.

Production systems do not have a logical semantics

# Computational Logic and Human Thinking

- CL as the Language of Thought (LOT)
- CL as a connectionist model of the mind
- Production systems as an alternative model of the Mind
- CL as a unifying framework

# Jonathan Baron “Thinking and Deciding” (Fourth edition, 2008)




“*Thinking* about actions,
beliefs and personal goals
can all be described in terms of
a common framework,

which asserts that
thinking consists of *search* and *inference*.

We *search* for certain objects and then *make inferences*
from and about the objects we have found.” (page 6)

## As Sherlock Holmes explained to Dr. Watson, in *A Study in Scarlet*:

“In solving a problem of this sort, the grand thing is to be able to reason backward. That is a very useful accomplishment, and a very easy one, but people do not practise it much. In the everyday affairs of life it is more useful to reason forward, and so the other comes to be neglected. There are fifty who can reason synthetically for one who can reason analytically.”

…….

“Most people, if you describe a train of events to them, will tell you what the result would be. They can put those events together in their minds, and argue from them that something will come to pass. There are few people, however, who, if you told them a result, would be able to evolve from their own inner consciousness what the steps were which led up to that result. This power is what I mean when I talk of reasoning backward, or analytically.”

Joseph M. Williams

# Style

## Toward Clarity and Grace

*With two chapters coauthored by*
Gregory G. Colomb

The University of Chicago Press
*Chicago and London*

## Contents

*Preface* ix

1 Causes 1

2 Clarity 17

3 Cohesion 45

4 Emphasis 67

5 Coherence I 81
*With Gregory G. Colomb*

6 Coherence II 97
*With Gregory G. Colomb*

7 Concision 115

8 Length 135

9 Elegance 153

10 Usage 169

*Notes* 199

*Acknowledgments* 201

*Index* 203

# Clarity

Not: The teacher gave the student a good mark.
She was happy.

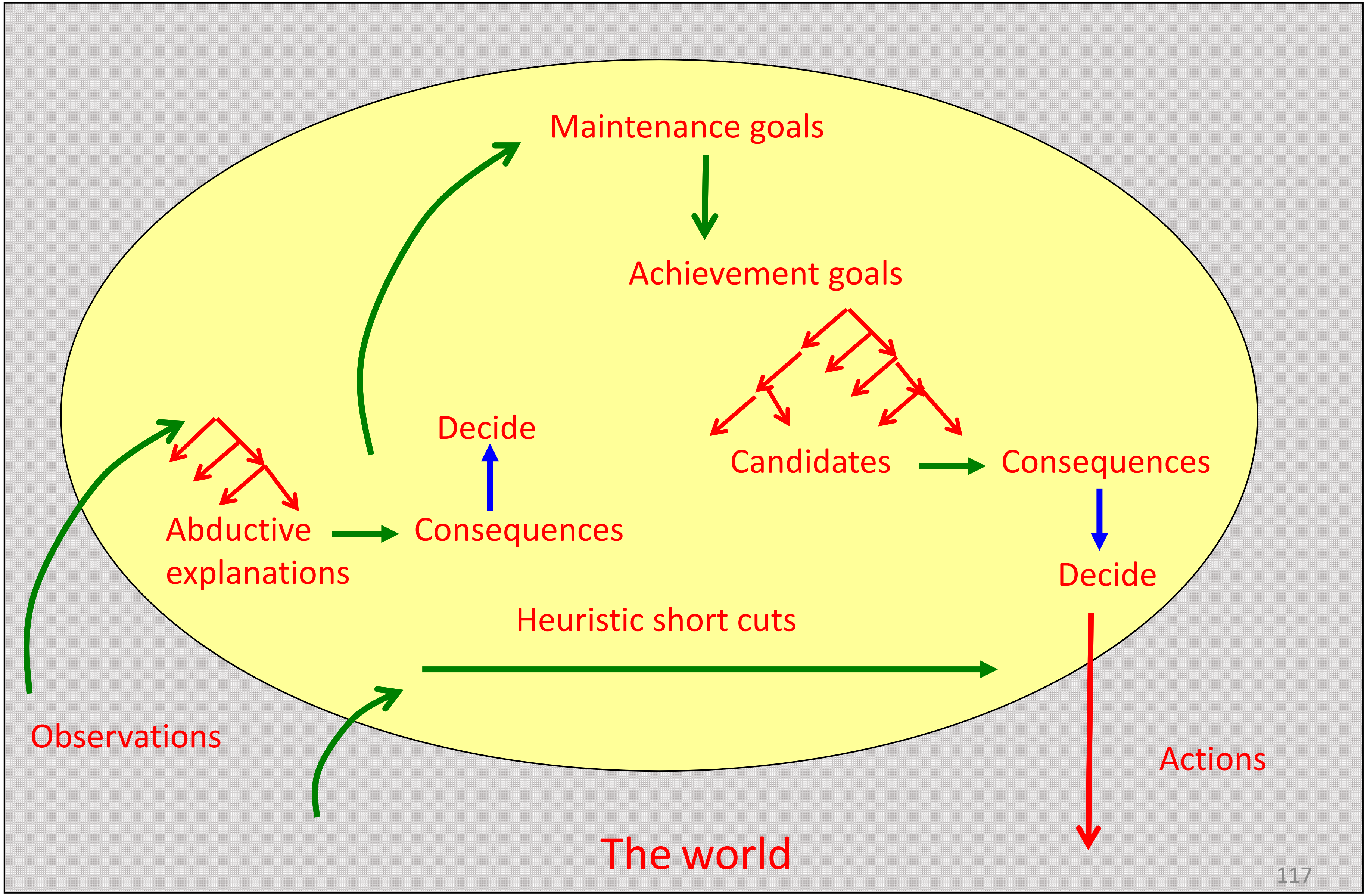
But: The teacher was happy.
Or: The student was happy.

?: Our lack of knowledge of the topic of the talk
prevented us from understanding it.

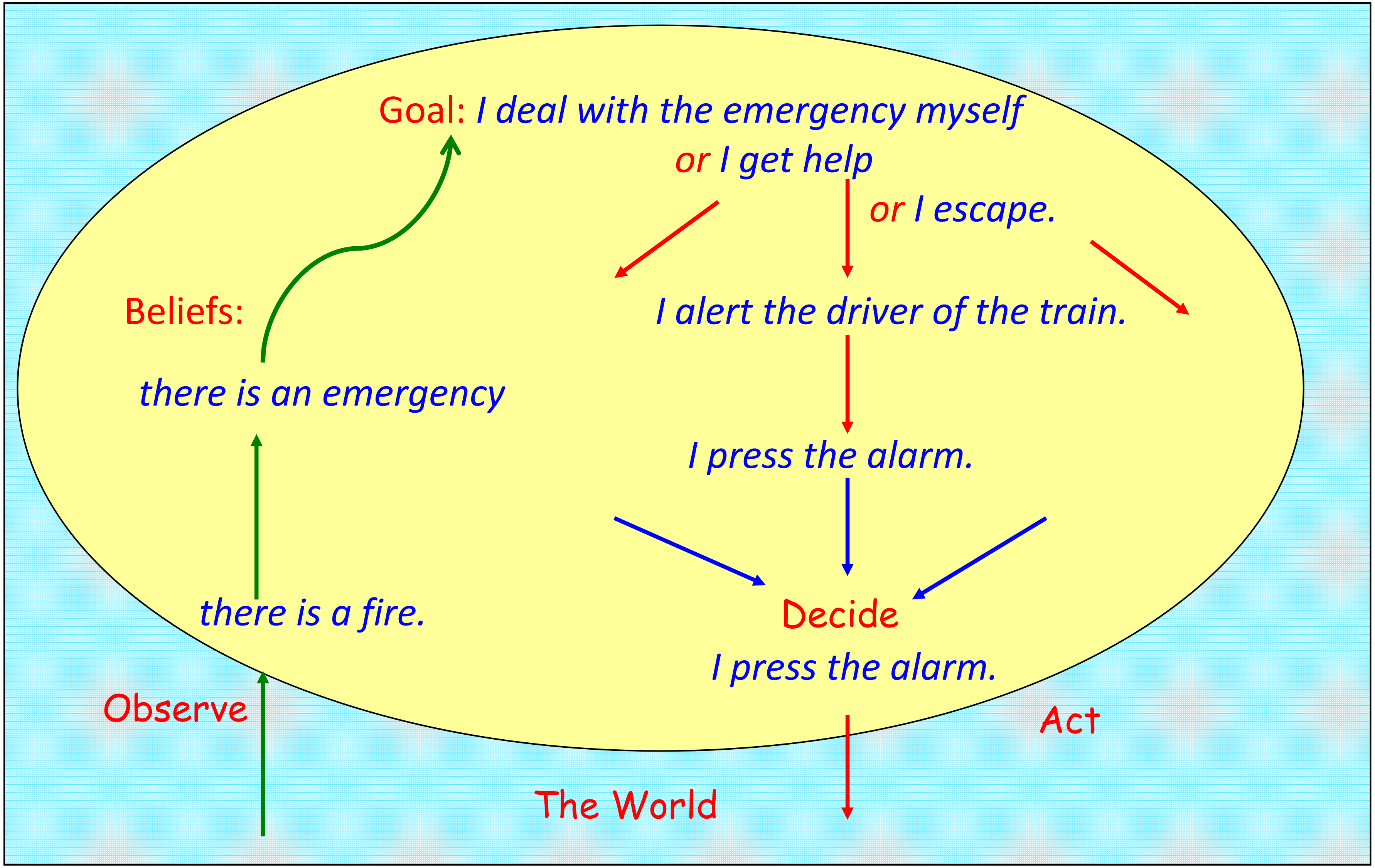
Or: Because we did not know the topic of the talk ,
we could not understand the talk.

Or: A person cannot understand a talk
if the person does not know the topic of the talk.
We did not know the topic of the talk.

# Thinking generates actions, to help an agent survive and prosper.

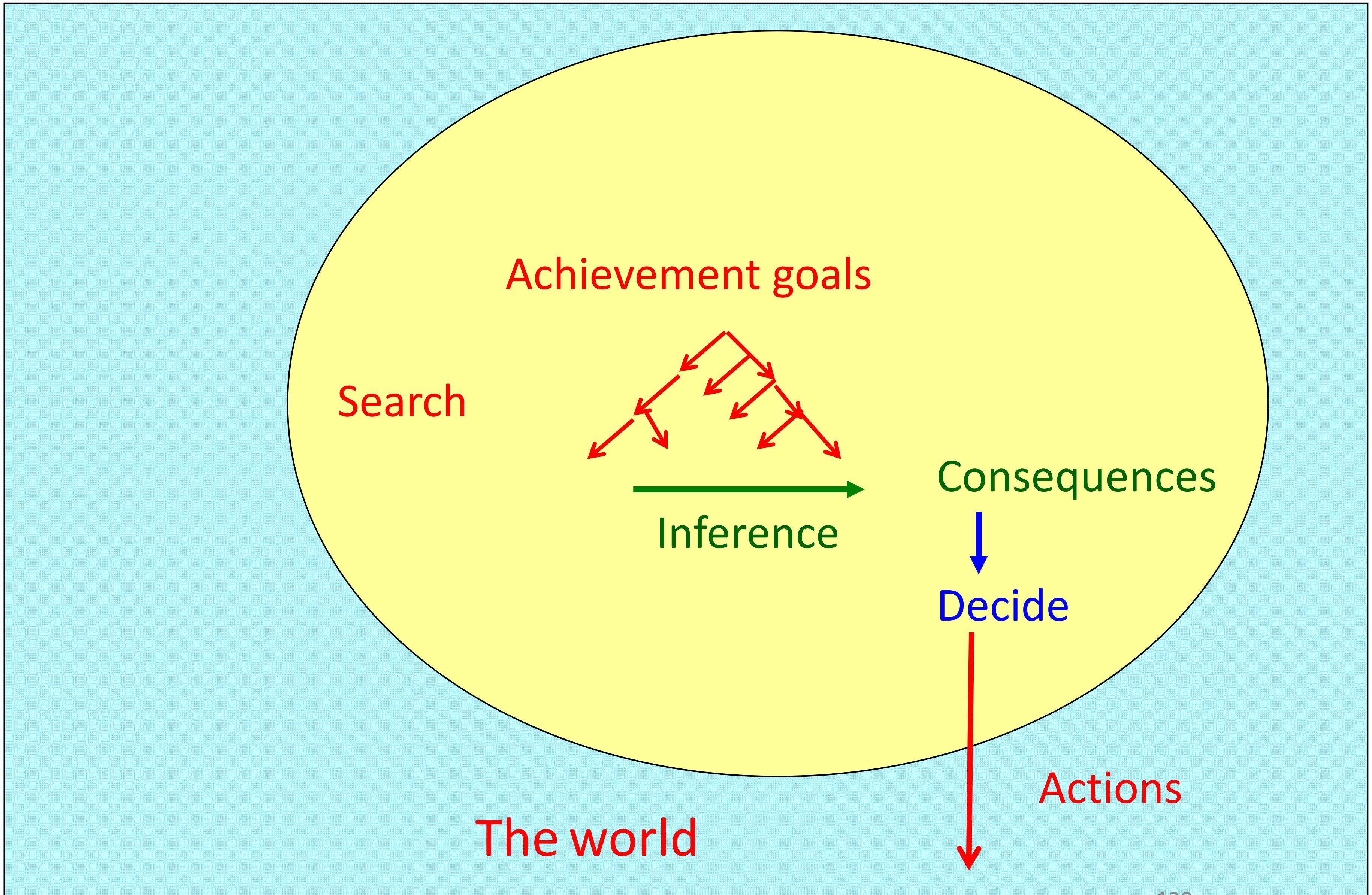


# The observe-think-decide-act agent cycle

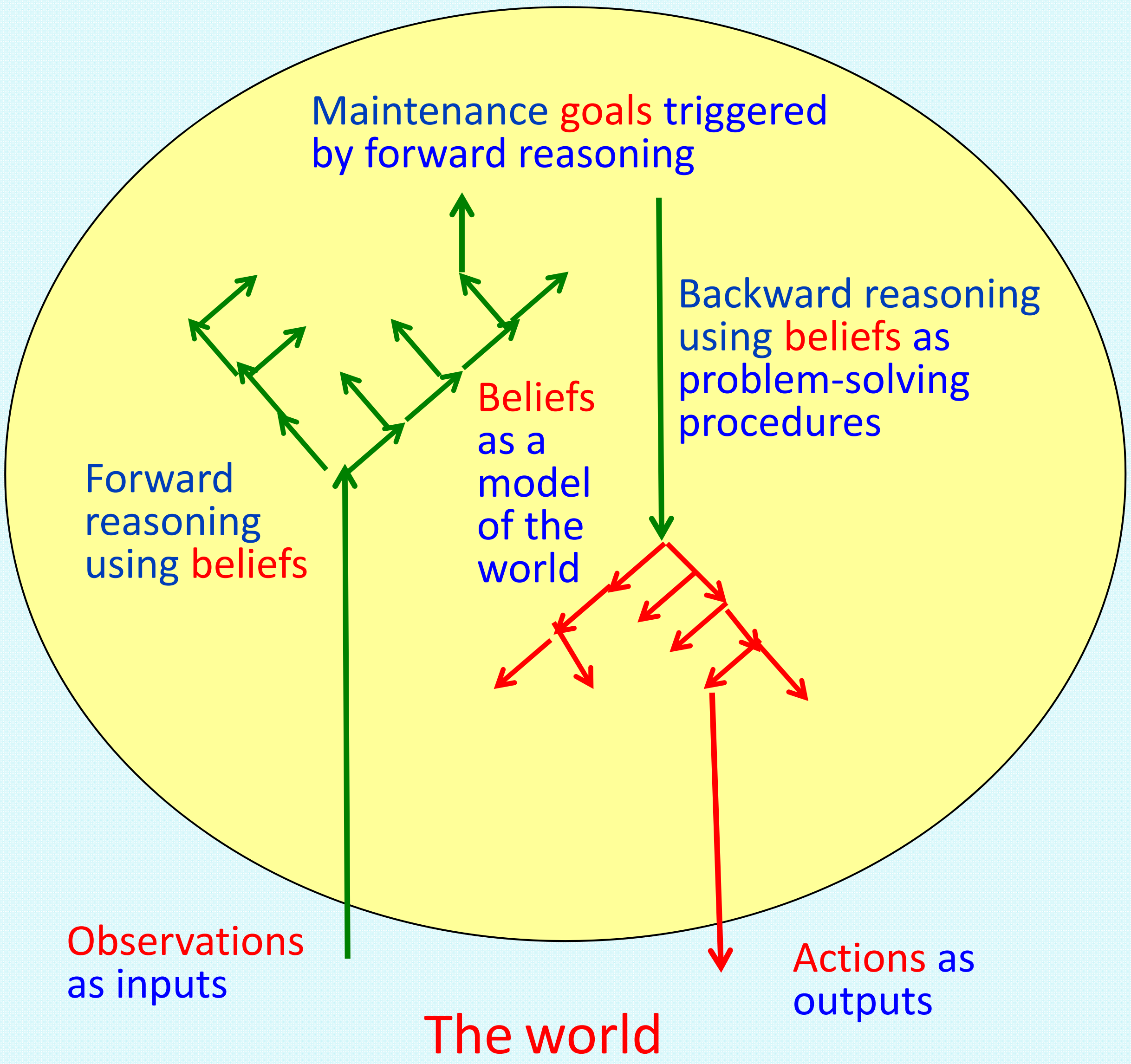


# The Logic of the agent cycle

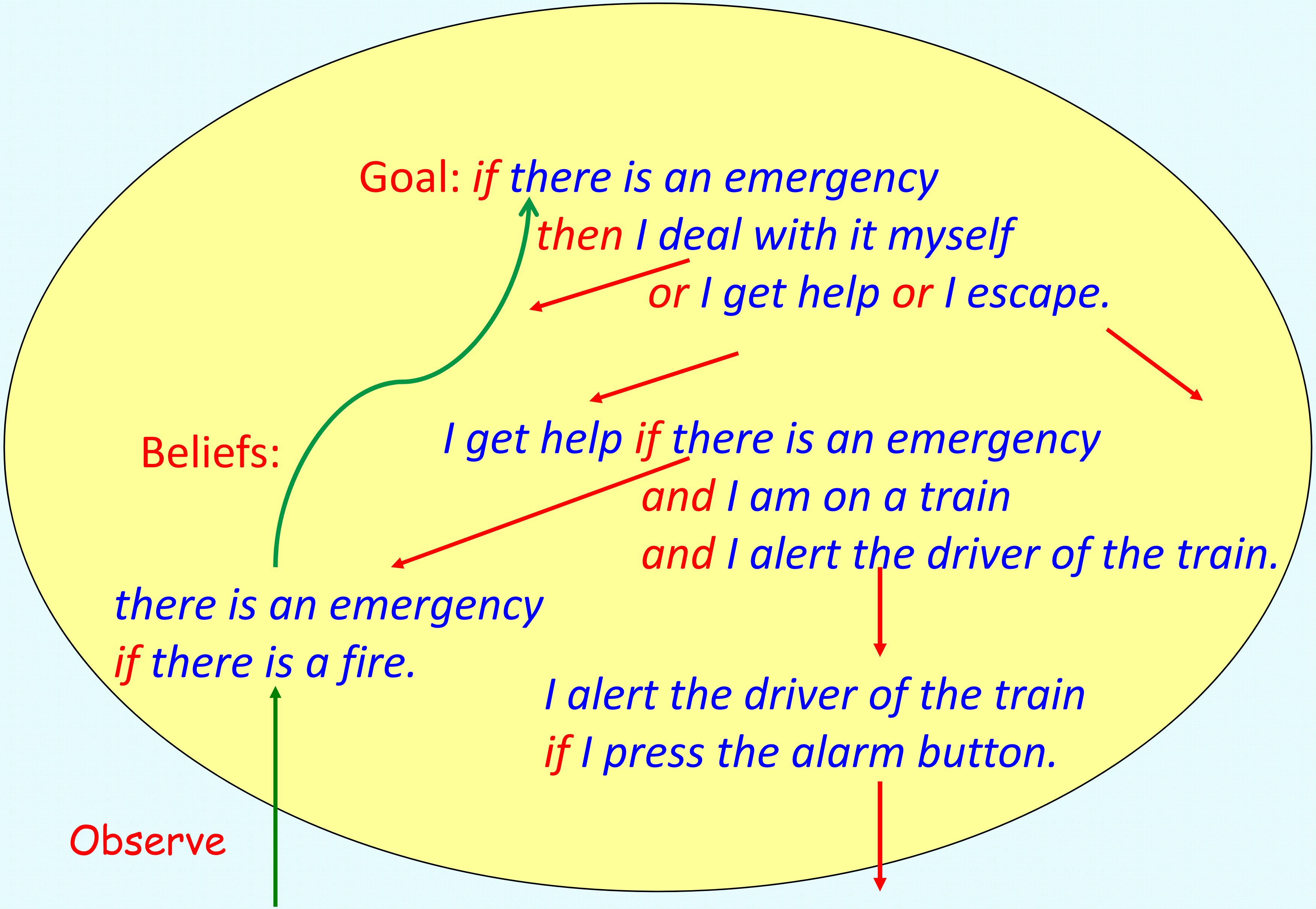
# Baron’s view of an intelligent agent



# Computational Logic as the Language of Thought of an intelligent agent



# Abductive Logic Programming



## The Heart of the Problem -
## What is the meaning of if A then B?

| | |
|---|---|
| Classical Logic: | If A is true then B is true. e.g. |
| | If X is a bird, then X can fly. |
| i.e. | bird(X) → fly(X). |
| equivalently | fly(X) ← bird(X) |
| | If X is mother of Y, then X is parent of Y. |
| I.e. | mother(X, Y) → parent(X, Y) |
| equivalently | parent(X, Y) ← mother(X, Y) |

## The Heart of the Problem -
## What is the meaning of if A then B?

Change of state. e.g. If A happens then do B.

If it is raining,
then cover yourself with an umbrella.

If you are hungry,
then buy food, cook the food, and eat the food.

If you want to go home for the weekend, and you have the bus fare,
then take the bus.

If you increase an employee's salary,
then increase the employee's manager's salary.

What is the meaning of if A then B?
A proposed solution in classical logic.

Add explicit time:
If A happens at time T, then you do B at time T+n.

If it is raining at time T,
then you cover yourself with an umbrella at time T+1.

If you are hungry at time T
then you buy food at time T+1, you cook the food at time T+2,
and you eat the food at time T+3.

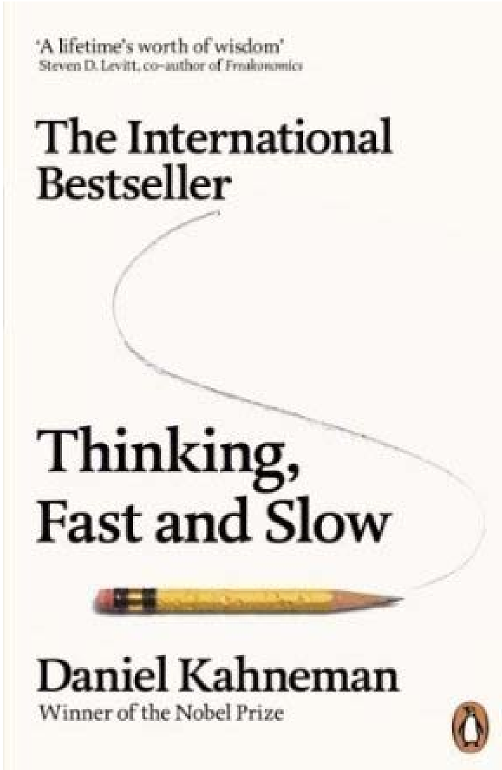
If you increase an employee's salary at time T1,
then you increase the employee's manager's salary at time T2
and T1 < T2 < T1 + 10.

# The dual process model combines two systems of thinking

System 1 operates automatically and quickly, with little or no effort and no sense of voluntary control.

System 2 allocates attention to the effortful mental activities that demand it, including complex computations.

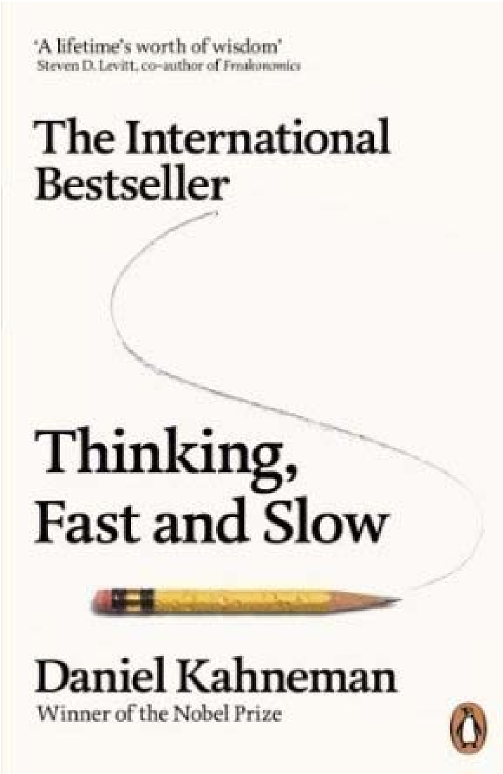
The operations of system 2 are often associated with the subjective experience of agency, choice, and concentration.



# The dual process model of thinking

System 1 “quickly proposes intuitive answers to judgement problems as they arise”,

System 2 “monitors the quality of these proposals, which it may endorse, correct, or override”.

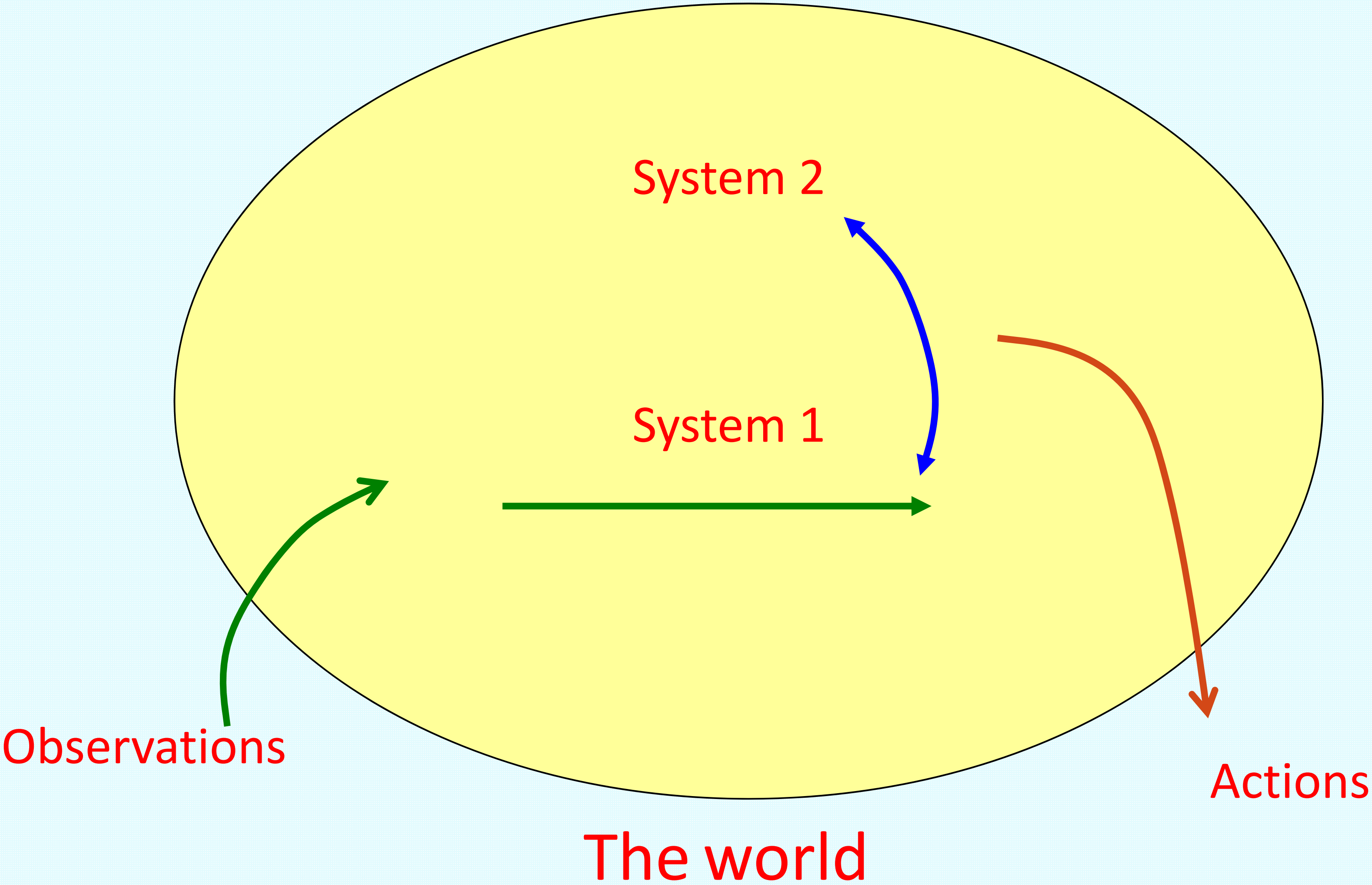



When system 1 runs into difficulty, it calls on system 2.

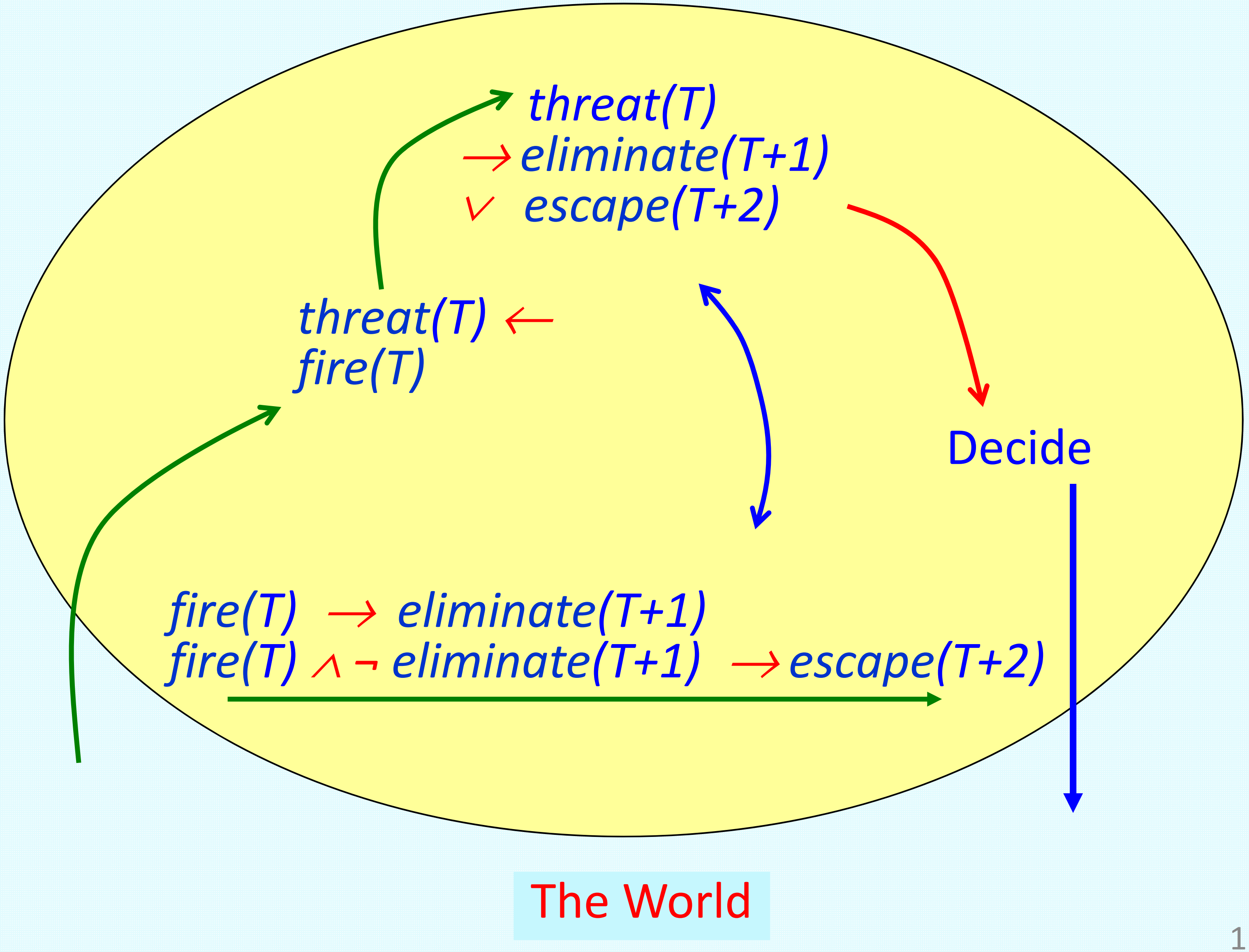
System 1 continuously generates suggestions for system 2.

System 2 is activated when an event is detected that violates the model of the world that system 1 maintains.

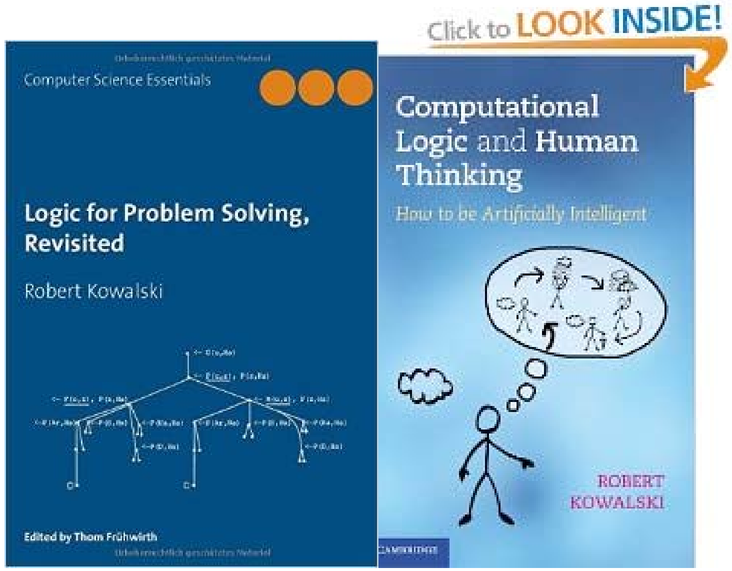
# The Dual Process Model of thinking



# The Dual Process Model viewed in logical terms

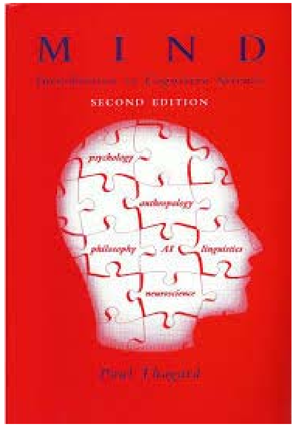


Computational Logic represents goals and beliefs in logical form




The task of an intelligent agent is to perform actions, to make its goals and observations true in the model of the world determined by its beliefs.

# Some authors confuse production rules and logic programs




Rules can be used to reason either *forward* or *backward*. Reasoning backward, a student might think that "To get home, I can take the highway, which requires taking the parkway, which requires taking Main Street, which requires getting a car." The goal is to get home, but the plan is constructed by considering a series of subgoals such as getting to the highway. Reasoning forward, the student might use inference akin to modus ponens to see that "Main Street gets me to the parkway, which gets me to the highway." Forward and backward reasoning both try to find a series of rules that can be used to get from the starting point to the goal, but they differ in the search strategy employed.

# Backward reasoning interprets logic programs as goal-reduction procedures

Backward reasoning interprets *C if A and B*
as a procedure for solving *C* by solving the subgoals *A* and *B*.

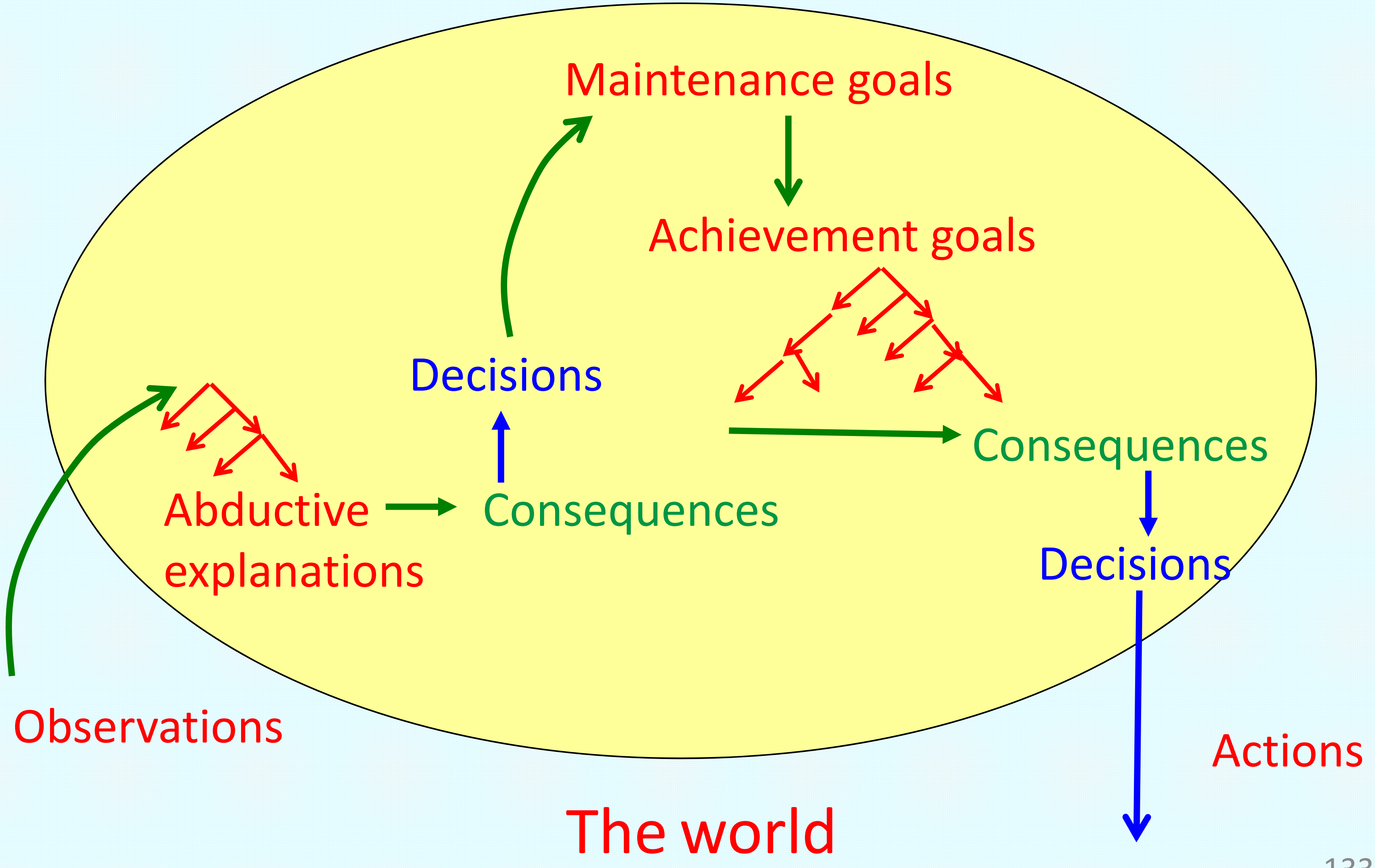
Backward reasoning interprets the logic program

*You go home*
*if you have the bus fare and you catch a bus*

as the procedure

*If you want to go home*
*and you have the bus fare,*
*then you can catch a bus.*

# CL = abductive logic programming (ALP) embedded in an observe-think-decide-act agent cycle.



## Abductive Logic Programming (ALP) combines goals and beliefs

Logic programs = beliefs B

*X = X*

*country(usa)* *country(canada)*

*adjacent(usa, canada)* *etc.*

Integrity constraints = goals G

*country(X) → colour(X, red) ∨ colour(X, blue) ∨ colour(X, yellow)*

*colour(X, C), colour(X, D) → C = D*

*colour(X, C), colour(Y, C), adjacent(X, Y) → false*

Abducible predicates = candidate hypotheses A

*colour(usa, red), colour(usa, blue), etc.*

## Abductive Logic Programming (ALP)

Logic programs = beliefs B

*X = X*

*country(usa)* *country(canada)*

*adjacent(usa, canada)* *etc.*

Integrity constraints = goals G

*country(X)* → *colour(X, red)* ∨ *colour(X, blue)* ∨ *colour(X, yellow)*

*colour(X, C), colour(X, D)* → *C = D*

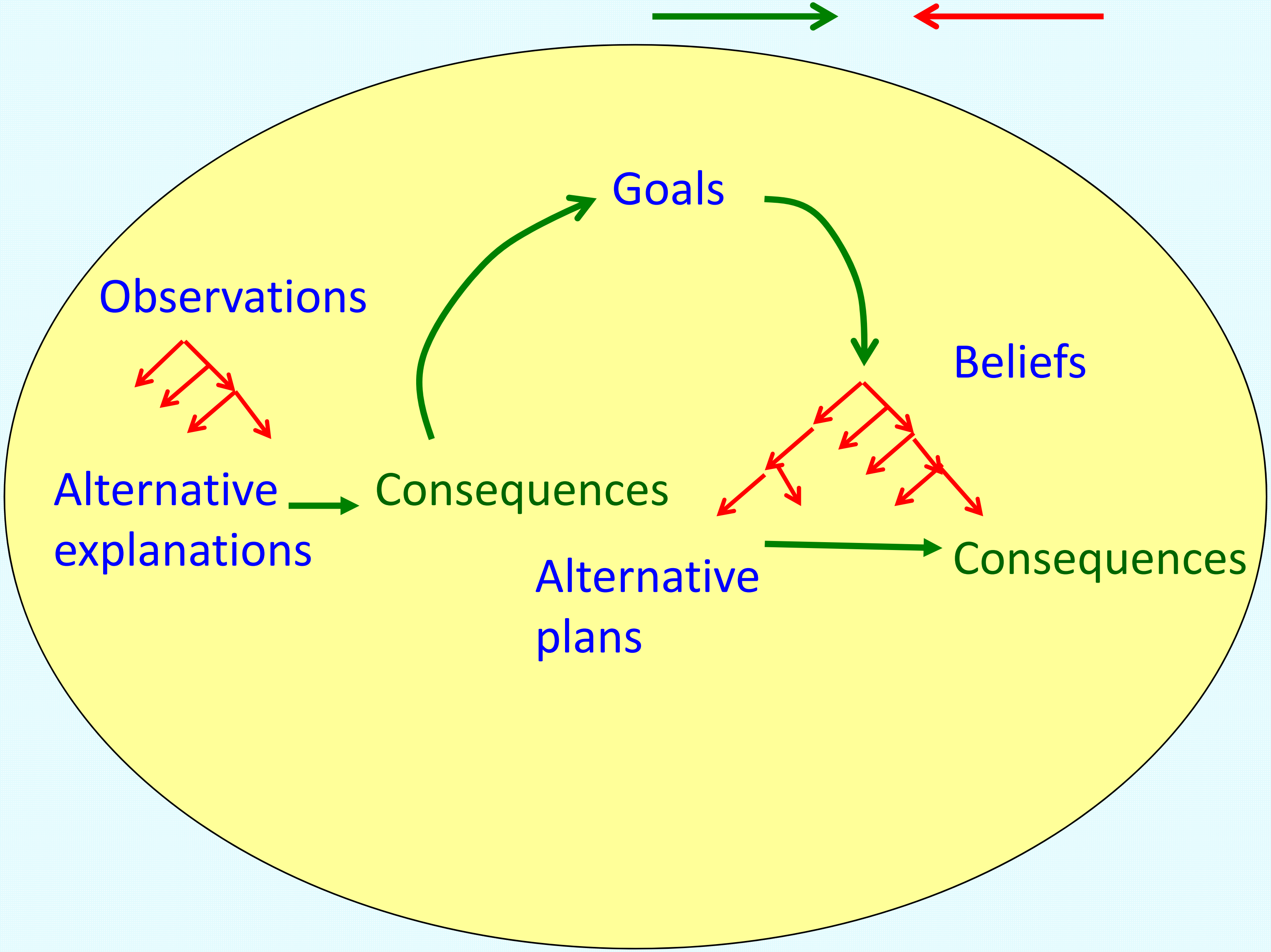
*colour(X, C), colour(Y, C), adjacent(X, Y)* → *false*

Abducible predicates = candidate hypotheses A

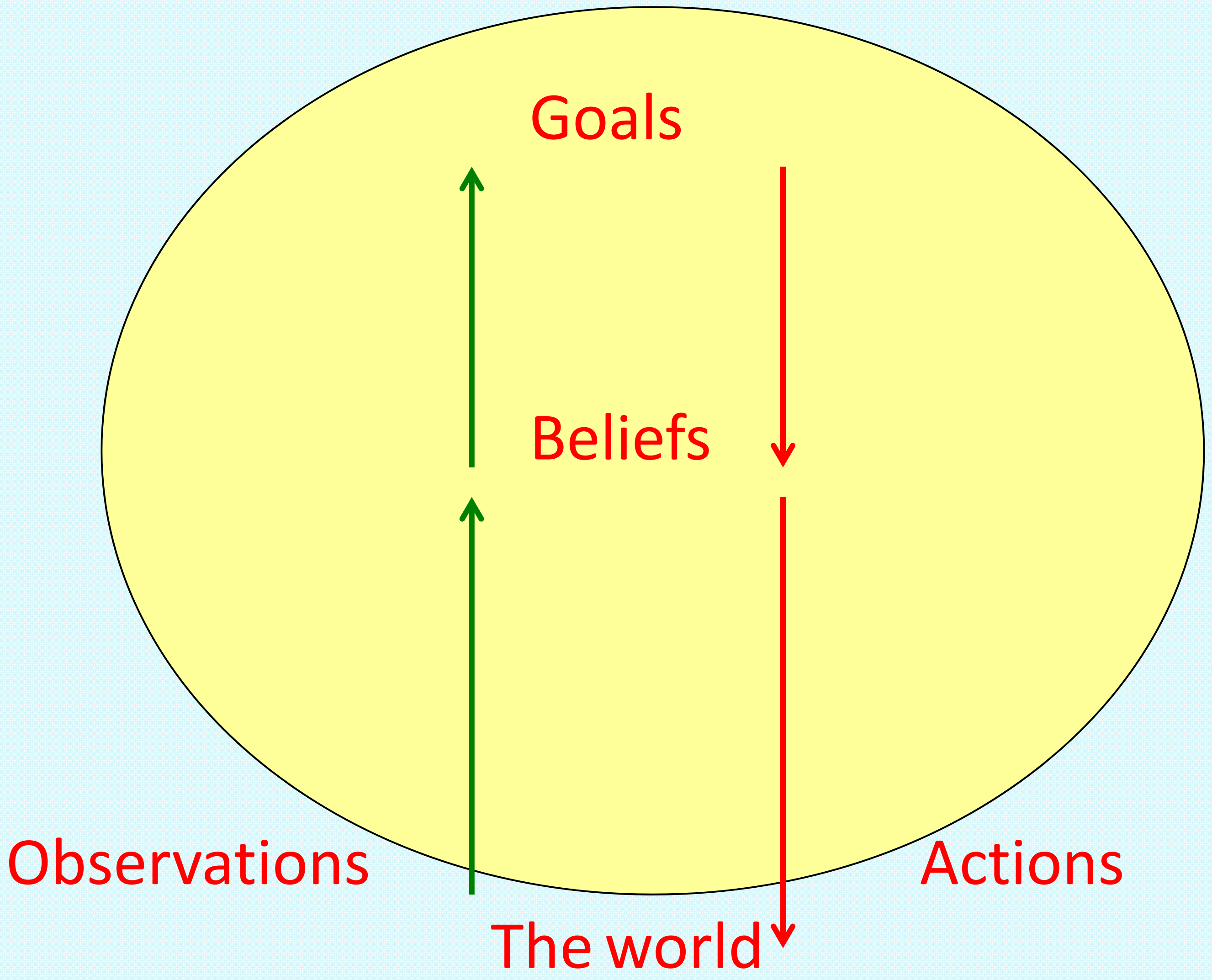
*colour(usa, red), colour(usa, blue), etc.*

The task is to generate hypotheses $\Delta \subseteq A$ such that *G* is true in the minimal model of $B \cup \Delta$.

# ALP Proof procedures combine forward and backward reasoning



An agent's task in life is to perform actions
to make its goals and observations true
in the model of the world determined by its beliefs



# Backward reasoning interprets logic programs as goal-reduction procedures

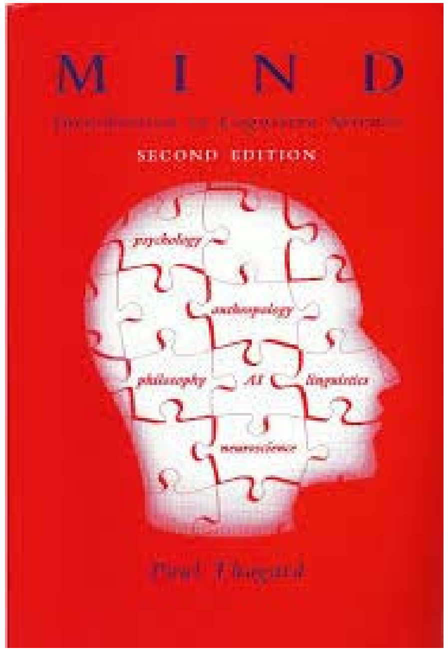
Backward reasoning interprets
the logic program

*the driver is alerted*
*if you press the alarm signal button.*

as the procedure

*If you want to alert the driver*
*then you press the alarm signal button.*

## Some authors are confused about the relationship between deduction and search




"In logic-based systems
the fundamental operation of thinking
is logical deduction,
but from the perspective of rule-based systems
the fundamental operation of thinking is search."

IF you drive on highway 1,
THEN you can get from university city to home city.

IF you take the parkway,
THEN you can get from university city to the highway.

IF you take a bus from the bus depot,
THEN you can get from university city to home city.
etc.

Logic program with explicit time:

$$
\begin{aligned}
reach(robot, X, T) &\leftarrow location(robot, X, T) \\
reach(robot, X, T2) &\leftarrow location(robot, Y, T1) \wedge \\
&\quad \text{not } X = Y \wedge \\
&\quad adjacent(Y, Z) \wedge \\
&\quad \text{not } location(car, Z, T1) \wedge \\
&\quad move(robot, Z, T1) \wedge \\
&\quad reach(robot, X, T2) \wedge T1 < T2
\end{aligned}
$$

Here *move(robot, Z, T1)* is a primitive (atomic) action.
If it succeeds, it initiates *location(robot, Z, T1)*
and it terminates *location(robot, X, T1) .*

The program is teleo-reactive.

# Abductive Logic Programming

Beliefs B: *there is smoke if there is a fire*
*there is an emergency if there is a fire*
*I get help if I press the alarm button*

Goal G: *if there is an emergency*
*then I deal with it myself or I get help or I escape*

Observation O: *there is smoke*

# CL/ALP combines forward and backward reasoning

# Abductive Logic Programming (ALP)

Beliefs B: *there is smoke if there is a fire*
*there is an emergency if there is a fire*
*I get help if I press the alarm button*

Goal G: *if there is an emergency*
*then I deal with it myself or I get help or I escape*

Observation O: *there is smoke*

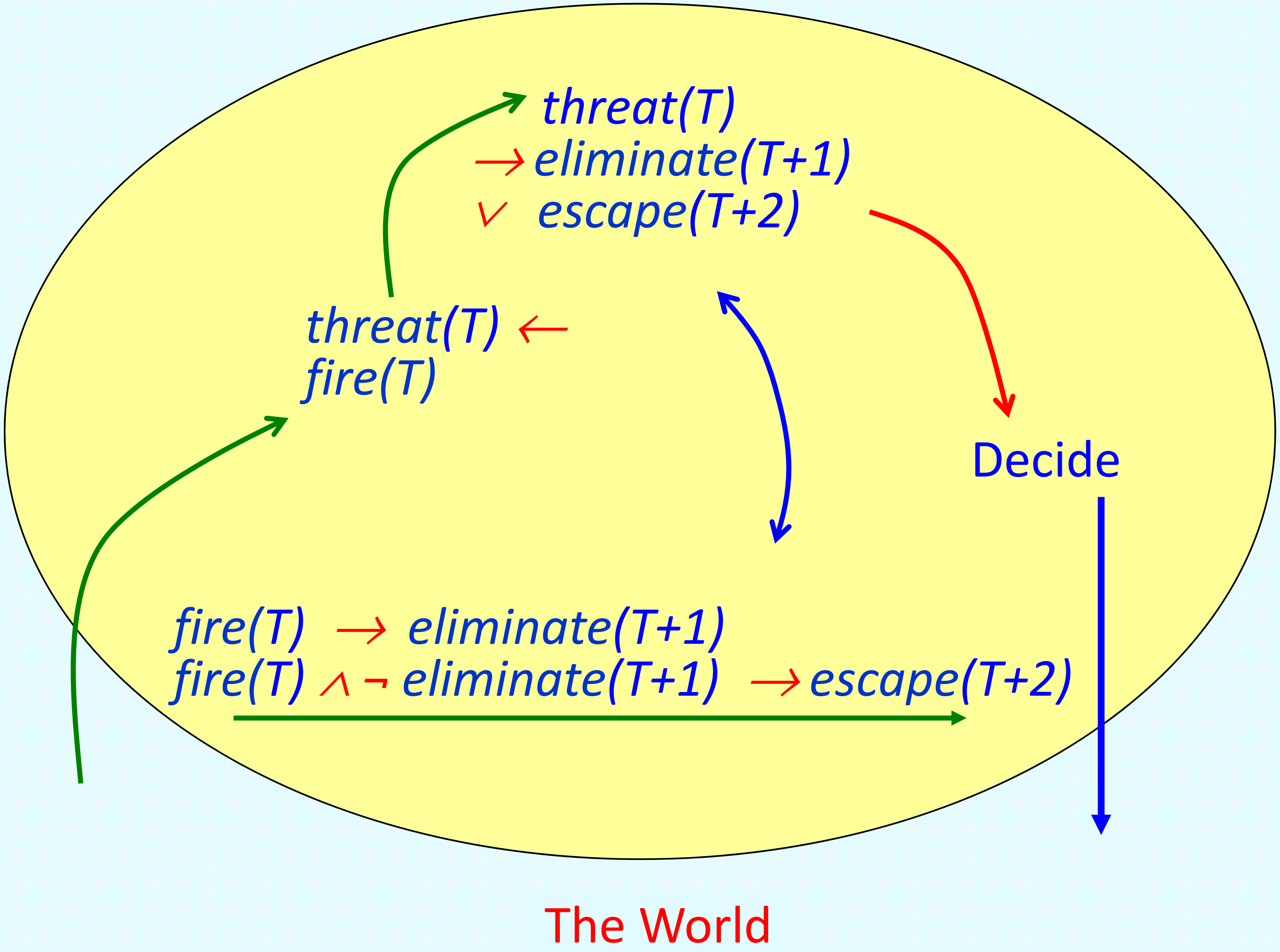
Assumptions Δ: *there is a fire*
*I press the alarm button*

G ∪ O is true in the model of the world determined by B ∪ Δ.

Abduction: *there is a fire* explains *O*

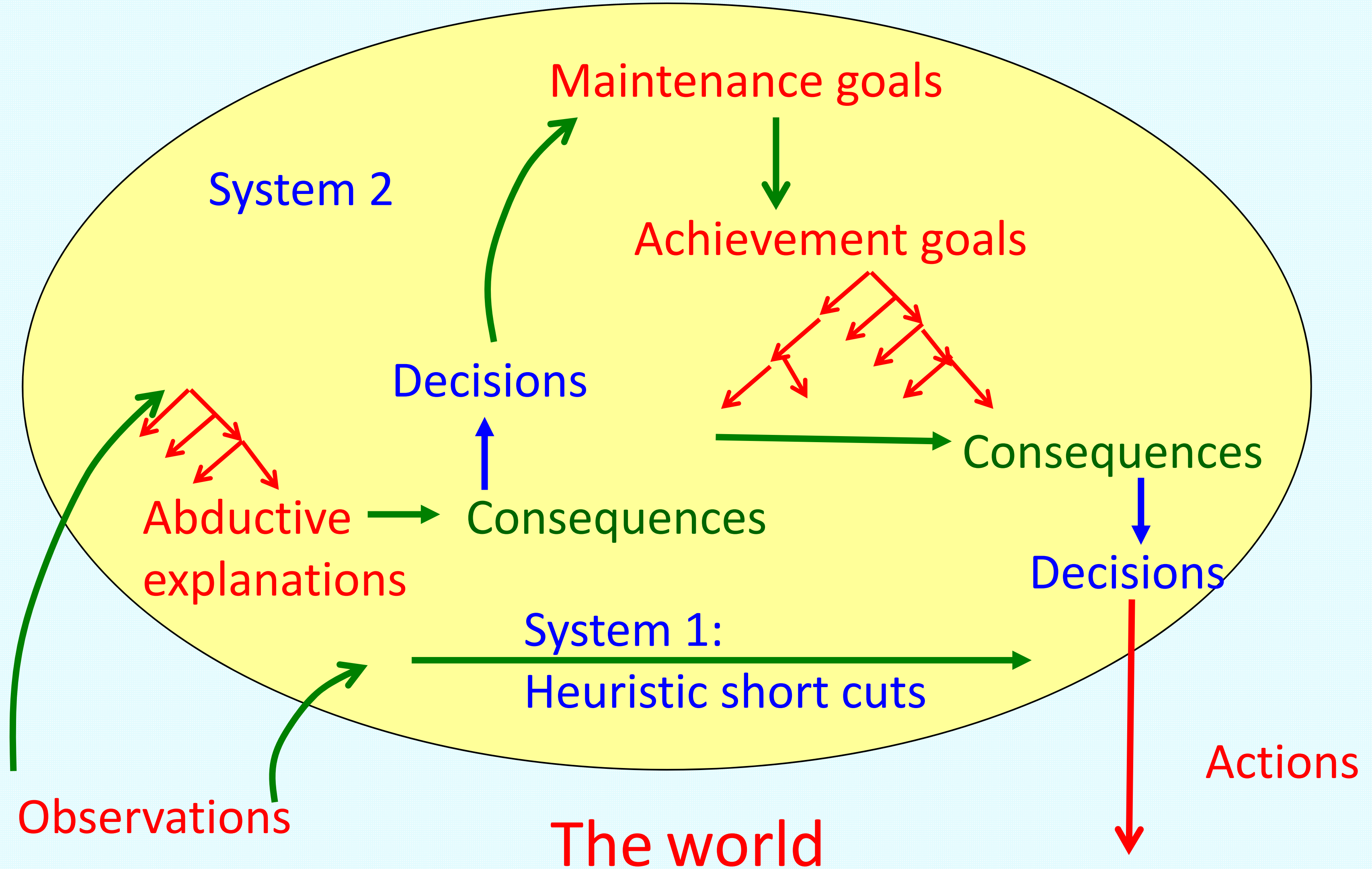
Planning: *I press the alarm button* achieves *G*

# The Dual Process Model viewed in logical terms



# CL/ALP combines forward and backward reasoning

# ALP agents as a unifying framework



Logic program: *you go home*
*if you have the bus fare,*
*and you catch a bus.*

More precisely and more generally:

$$
\begin{aligned}
& \quad at(Agent, Destination, T2) \\
\leftarrow\ & at(Agent, Location, T1) \wedge \\
& have(Agent, Money, T1) \wedge \\
& busRoute(Bus, Location, Destination, Fare) \wedge \\
& Fare < Money \wedge \\
& take(Agent, Bus, Location, T1) \wedge \\
& arrives(Bus, Destination, T2) \wedge\ T1 < T2
\end{aligned}
$$

## AgentSpeak(L):

```
+!location(robot,X):location(robot,X) <- true.   (P2)

+!location(robot,X):location(robot,Y) &
                    (not (X = Y)) &
                    adjacent(Y,Z) &
                    (not (location(car, Z)))
                              <- move(Y,Z);
                                 +!location(robot,X). (P3)
```

Logic program with explicit time:

$$
\begin{aligned}
reach(robot, X, T) &\leftarrow location(robot, X, T) \\
reach(robot, X, T2) &\leftarrow location(robot, Y, T1) \wedge \\
&\quad not\ X = Y \wedge \\
&\quad adjacent(Y, Z) \wedge \\
&\quad not\ location(car, Z, T1) \wedge \\
&\quad move(robot, Z, T1) \wedge \\
&\quad reach(robot, X, T2) \wedge T1 < T2
\end{aligned}
$$