

Computational Logic and Human Thinking

Robert Kowalski
Imperial College London

based on *Computational Logic and
Human Thinking – How to be Artificially Intelligent*
Cambridge University Press (June 2011)

Download at <http://www.doc.ic.ac.uk/~rak/>

Computational Logic and Human Thinking

How to be Artificially Intelligent



ROBERT
KOWALSKI

CAMBRIDGE

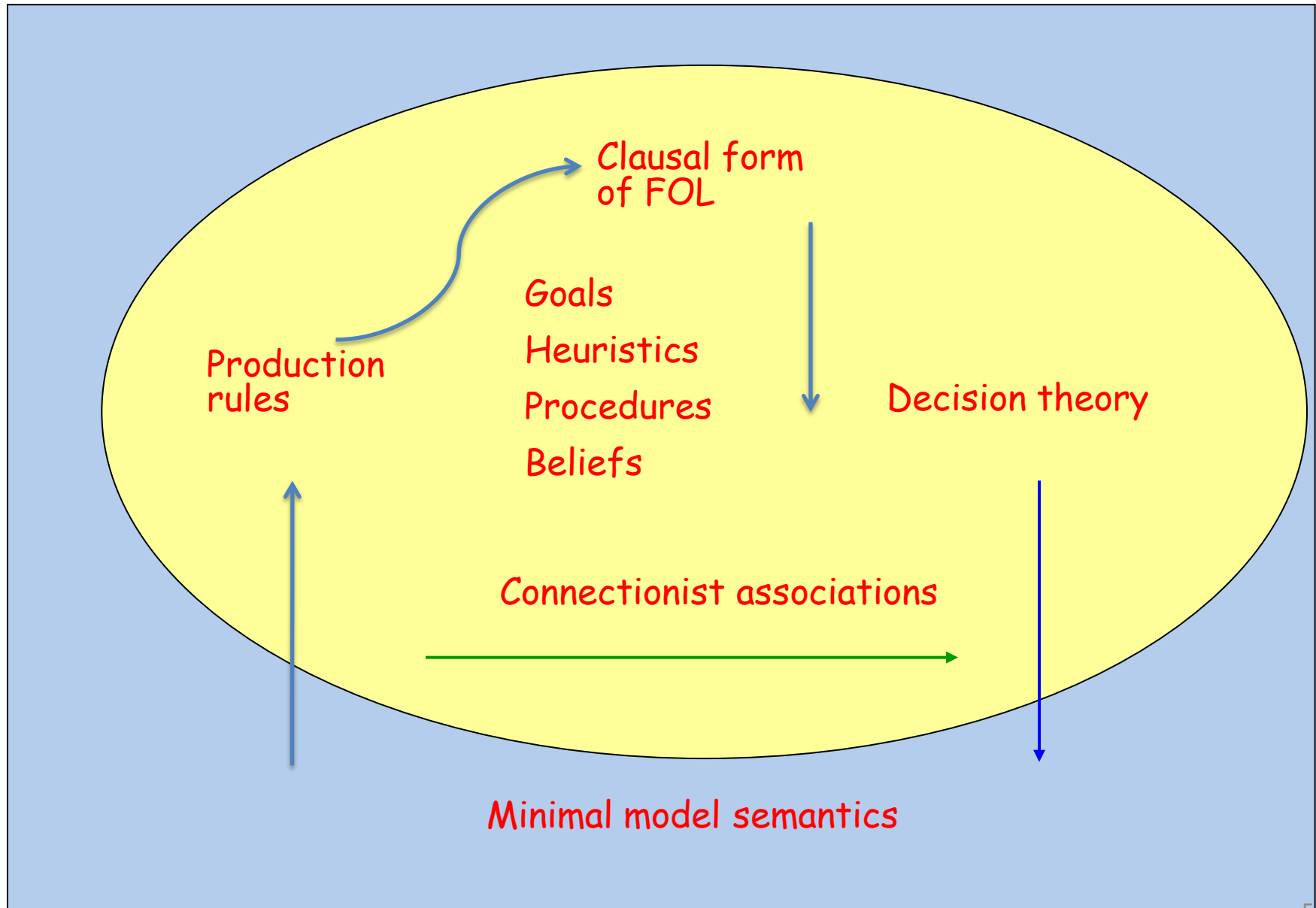
Overview – Kyoto 2012

Lecture 1	Overview and Chapter 1 (London Underground)	Oct 3
Lecture 2	Chapter A1 The Syntax of Logical Form	Oct 10
Lecture 3	Chapter 2 The Psychology of Logic	Oct 17
Lecture 4	Chapters A2 Truth and A3 Forward and Backward Reasoning	Oct 24
Lecture 5	Chapter 4 Search	Oct 24
Lecture 6	Chapter 5 Negation as Failure	Oct 31
Lecture 7	Chapter 6 How to Become a British Citizen	Nov 7
Lecture 8	Chapter 7 The Louse and the Mars Explorer	Nov 14
Lecture 9	Chapter 8 Maintenance Goals as the Driving Force of Life	Nov 14
Lecture 10	Chapter A5 The Resolution Rule	Nov 21
Lecture 11	Chapter 10 Abduction	Nov 28
Lecture 12	Chapter 11 The Prisoner's Dilemma	Dec 5
Lecture 13	Chapter A4 Minimal Models and Negation	Dec 5
Lecture 14	Conclusions	Dec 12

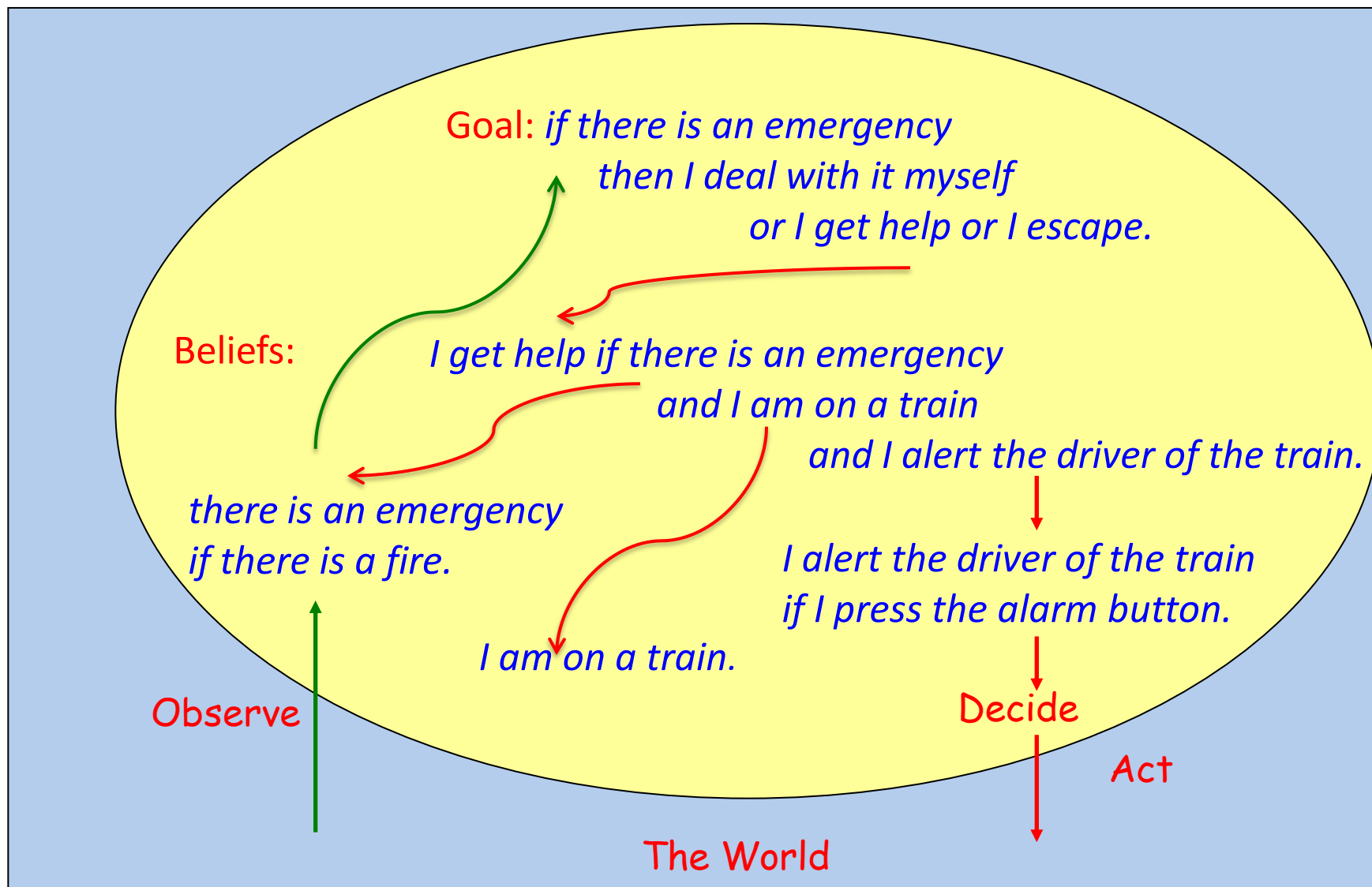
AI tools and techniques

- can be reconciled and combined
 - Logic
 - Procedural representations
 - Heuristics
 - Production Systems
 - Bayesian networks
 - Connectionism
- can help people
 - make better decisions
 - communicate more effectively with humans and machines.

AI tools and techniques can be reconciled and combined



An Agent on the London Underground



Complex thinking and decision-making can be compiled into more efficient, lower-level maintenance goals, heuristics (or input-output associations)

For example:

*if there is a fire
and I am on a train
and I can not deal with the fire myself
then I press the alarm button.*

Cognitive Psychology

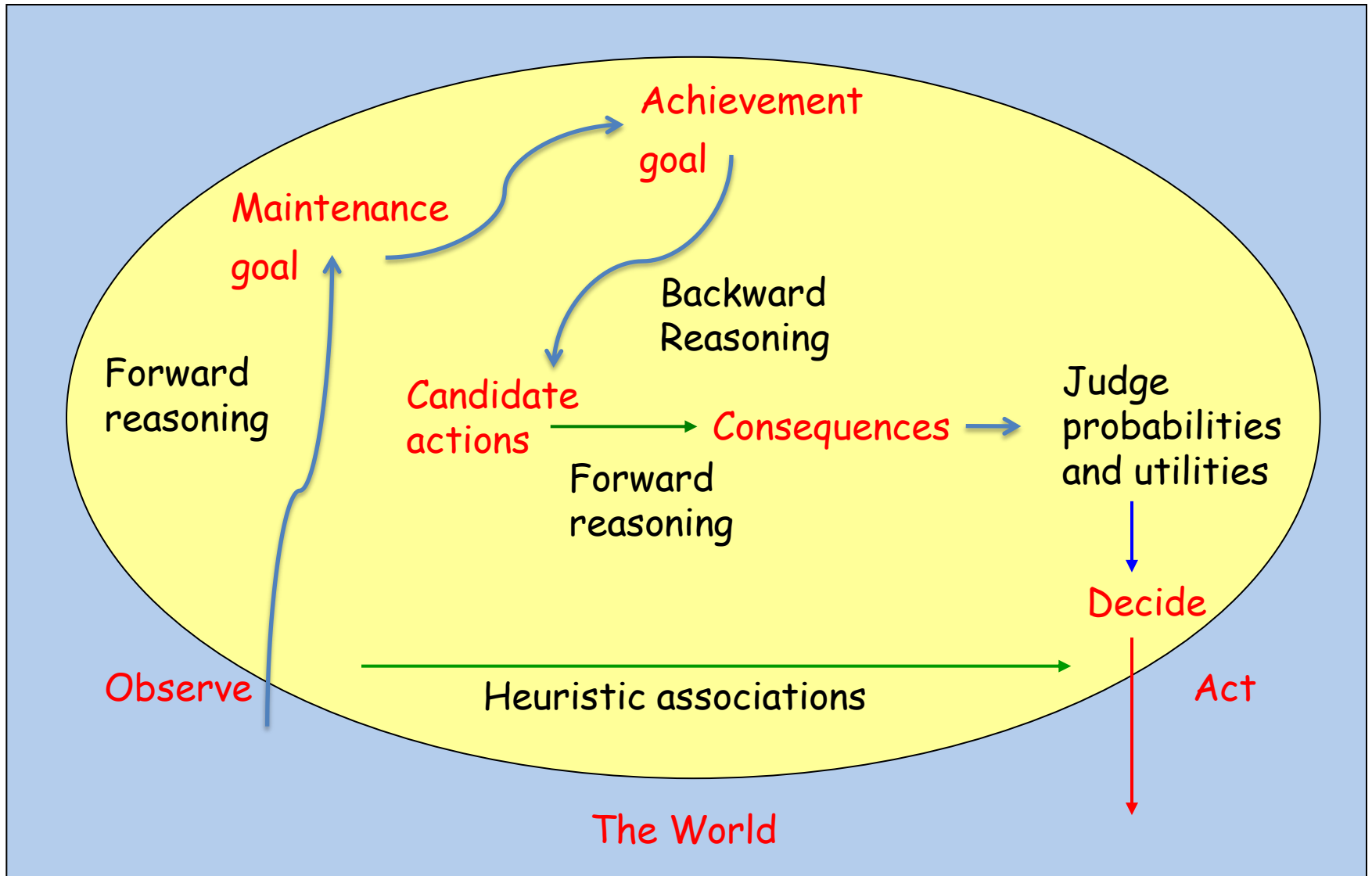
The dual process models of human thinking:
Lower-level heuristics and
higher-level thinking and deciding
can be combined.

As Kahneman and Frederick (2002) put it:

the **intuitive, subconscious level** “quickly proposes intuitive answers to judgement problems as they arise”,

while the **deliberative, conscious level** “monitors the quality of these proposals, which it may endorse, correct, or override”.

Computational Logic (CL) Agent: operational semantics



A Connectionist implementation of CL



BDI agents in AI (Beliefs Desires Intentions)
can be understood in CL agent terms

Agents use their *beliefs* to attain their *desires* by generating *intentions*, which are selected plans of actions. (Selection of actions/plans is the subject of *decision theory/analysis*)

In CL agents, beliefs and desires (or goals) are both represented in *simplified logical form*.

Beliefs are represented as *logic programs*.

Goals are represented in *first-order logic (FOL)*.

Goals and Beliefs

Goals (in FOL)

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

Beliefs (as a logic program)

*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.

I alert the driver of the train if I press the alarm button.

I am on a train.

CL as the Language of Thought (LOT)

In the philosophy of language, there are three schools of thought :

The LOT is a private, language-like representation, which is independent of public, natural language.

The LOT is a form of public, natural language.
The natural languages that we speak influence the way we think.

The LOT does not exist.

In CL agents, computational logic serves as an agent's private LOT, independent of any public natural language.

CL as the LOT

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

It follows that:

If you want to identify the LOT, then you should study communications that are easy to understand.

If you want your communications to be easy to understand, then you should express them in a form that is close to the thoughts that you want to convey.

Computational logic and natural language: The Emergency Notice on the London underground

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 Logic of the London Underground Notice

The first sentence

Press the alarm signal button **to** alert the driver.

is a procedural representation of a logic program:

the driver is alerted

if you press the alarm signal button.

In general, a procedure of the form:

Do plan to achieve goal

can be represented in the logic programming form:

goal if plan.

The Logic of the London Underground Notice

The second sentence

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

is **ambiguous**, and one of its conditions has been **omitted**:

*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 Logic of the London Underground Notice

The logic of the third sentence

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

is two sentences, say:

*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.*

The Logic of the London Underground Notice

The fourth sentence

There is a 50 pound penalty **for** improper use.

is also a conditional, but in disguise:

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

The syntax of goals and beliefs

Beliefs: Logic programming *clauses* have the form:

conclusion if condition₁ and condition₂ and condition_n

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

The syntax of goals and beliefs

Goals: clauses of the form:

*If condition₁ and condition₂ and condition_n
then conclusion₁ or conclusion₂ or conclusion_m*

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

*it is not the case that
condition₁ and condition₂ and condition_n*

CL agents - operational semantics

Reason forwards from observations and forwards and backwards from beliefs, to determine whether some instance of the conditions of a goal

*If condition₁ and condition₂ and condition_n
then conclusion₁ or conclusion₂ or conclusion_m*

is *true*. Goals understood in this way are called *maintenance goals*.

Derive the corresponding instance of the conclusion of the goal

conclusion₁ or conclusion₂ or conclusion_m

as an *achievement goal*, to make *true*.

Reason backwards from achievement goals, reducing goals to subgoals, searching for a plan of actions that solves the goals.

CL agents - model-theoretic semantics

Beliefs describe the world as the agent sees it.

Goals describe the world as the agent would like it to be.

In deductive databases:

Beliefs represent data.

Goals represent queries and integrity constraints.

Given beliefs B , goals G and observations O ,

The purpose of an agent's life is to generate a set Δ
of actions and assumptions about the world such that:

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

Abduction

Classical abduction is the task of generating assumptions Δ to explain observations O .

For example, if instead of observing fire,
I observe *there is smoke*, and
I believe *there is smoke if there is a fire*.

Backwards reasoning from the observation generates an assumption *there is a fire*.

Forward and backward reasoning then continue as before, to generate a plan of actions to deal with the emergency.

Abductive logic programming (ALP) combines abduction and CL agents

There can be several, alternative Δ that, together with B , make G and O both *true*.

The challenge is to find the best Δ within the computational resources available.

In **classical decision theory**, the value of an action is measured by the expected utility of its consequences.

In **philosophy of science**, the value of an explanation is measured similarly in terms of its probability and explanatory power. (The more observations explained the better.)

In ALP/CL agents, the same measure can be used to evaluate both candidate actions and candidate explanations.

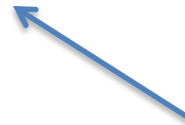
In both cases, candidate assumptions in Δ are evaluated by using forward reasoning to generate consequences of the assumptions in Δ .

Reasoning with uncertainty

ALP/CL is compatible with probability

The general pattern of cause and effect:

a particular outcome happens
if I do a certain action
and the world is in a particular state.



uncertain

David Poole [Journal of Artificial Intelligence, 1997] has shown that associating probabilities with assumptions gives ALP the expressive power of Bayesian networks.

Uncertainty

For example:

*You will be rich if you buy a lottery ticket
and your number is chosen.*

*It will rain if you do a rain dance
and the gods are pleased.*

You can control your own actions
(like *buying a ticket* or *doing a rain dance*).

But you cannot always control the state of the world
(*your number is chosen* or *the gods are pleased*).

You might be to judge its probability (*one in a million?*).

Classical decision theory makes unrealistic simplifying assumptions

Uncertainty is one of the complications contributing the problem of deciding what to do.

To reduce this complexity, classical decision theory makes the simplifying assumption that all of the alternatives to be decided between are given in advance.

For example, if you are looking for a new job, classical decision theory would assume that all of the job options are given, and it would focus on the problem of deciding which option is most likely to give the best outcome.

Smart choices – a better decision theory

But as [Keeney, 1992; Hammond *et al.*, 1999; Carlson *et al.*, 2008]] and other decision analysts point out,

the assumption that all alternatives are given in advance is not only unrealistic as a descriptive model of human decision making, but it is also unhelpful as a normative model.

To make a good decision between alternatives, it is necessary first to establish the goals (or problem) that motivate the alternatives. These goals might be generated explicitly by higher-level thinking or they might be hidden implicitly in lower-level heuristic rules.

Smart choices involve creative generation of alternatives

For example, you might receive an offer of a new job when you are not looking for one, and you may be tempted to limit your options simply to deciding between accepting or rejecting the offer.

If you step back from the temptation, and think about the broader context of your goals, then you might generate other alternatives, like perhaps using the job offer to negotiate an improvement in the conditions of your current employment.

Conclusions

ALP/CL agent model arguably

reconciles and combines FOL, production systems, decision theory, connectionism, and probability.

can be used by ordinary people to improve their own human intelligence.

can help people communicate more effectively with humans and machines.

can help people make smarter choices.

Homework -

One page English essay of about 500-600 words, giving:

Your background, as relevant to this course. For example, have you had a course in logic before? Prolog? Your mathematical background? Etc.

Your intellectual and professional plans for the future. Do you have alternative plans? Do the outcomes of the alternatives depend on outside influences outside your control? Do the alternatives have different utilities or probabilities?

How do you think this course might be useful to you?

Try to express yourself as logically as you can.

Do not worry too much about grammar, but do the best you can.

Computational Logic and Human Thinking

Lecture 2 – The syntax of logical form – Chapter A1

Chapter A1

Some examples (not in the book):

Complex events and complex actions

Simple conversation

Blocks world

Dining philosophers

Atomic formulas (or atoms) are
the basic building blocks of symbolic logic

Atoms in logic are collections of *terms*, like “train”, “ driver” and “station”, held together by *predicate symbols*, like “in” or “stop”.

Predicate symbols are like verbs in English, and terms are like nouns or noun phrases.

the driver stops the train

predicate symbol

arguments

stop(driver, train)

constant terms

Alternative representations are possible

Propositional logic (atoms have no internal structure)
A predicate symbol with zero arguments (0-order logic)

happens-stop-driver-train

Meta-logic or higher-order logic
(predicates or predicate symbols are arguments of higher-level predicates)

happens(stop, driver, train)

Meta-logic or higher-order logic with reification
(events and other first-order objects are made into constants)

happens(event-0014)
type(event-0014, stop)
agent(event-0014, 007)
object(event-0014, the-flying-scotsman)
isa(007, train-driver)
isa(the-flying-scotsman, train)

Terms “denote” (or represent) individuals

Constants like: 007

Variables, which stand for whole classes of individuals, like: $X + Y = Y + X$

Variables start with an **upper case letter**, like X or Y .

Constants, function symbols and predicate symbols start with a **lower case letter**.

The opposite convention is also common. i.e. $x + y = y + x$.

Complex terms can be constructed using **function symbols**:

<i>mother of X</i>	written	<i>mother(X)</i>
$2 + 3$	written	$+(2, 3)$

However, functions can be written as relations, For example:

$mother(cain) = eve$	can be written	$mother(cain, eve)$
----------------------	----------------	---------------------

$+(2, 3) = 5$	can be written	$+(2, 3, 5)$
---------------	----------------	--------------

Function symbols make it possible to name an infinite number of individuals with a finite vocabulary.

The *natural numbers* $0, 1, 2, \dots$ can be represented by

$0, s(0), s(s(0)), \dots$

The function symbol s is called the *successor function*.

The term $s(X)$ represents $X+1$.

$2 + 3 = 5$ can be represented by $+(s(s(0)), s(s(s(0))), s(s(s(s(s(0)))))))$

Terms that contain no variables are called *ground terms*.

Conditionals and logic programs

A *conditional* is a sentence of the form $A \rightarrow B$, where A and B are sentences. However, we use the expression *conditional* to refer to sentences that may contain variables.

Logic programs are a special case of sets of conditionals (also called *clauses*):

$$C_1 \wedge \dots \wedge C_n \wedge \neg D_1 \wedge \dots \wedge \neg D_m \rightarrow E$$

equivalently $E \leftarrow C_1 \wedge \dots \wedge C_n \wedge \neg D_1 \wedge \dots \wedge \neg D_m$

where the *conclusion* E is an atomic formula, the *conditions* C_i are atomic formulas, and the *conditions* $\neg D_j$ are the negations of atomic formulas.

n and m can be 0.

If m is 0, then the conditional is called a *definite clause*.

If $n+m$ is 0, then $E \leftarrow$ (or $\rightarrow E$) is normally written simply as E .

Expressions connected by \wedge are called *conjunctions*.

Expressions connected by \vee are called *disjunctions*.

Variables and quantifiers

Predicate symbols and constant symbols link clauses together.
Variables link parts of the same clause together.

$amazing(X) \leftarrow can_fly(X)$
i.e. $\forall X (amazing(X) \leftarrow can_fly(X))$
anything that can fly is amazing.

$amazing(X) \leftarrow can_fly(Y)$
i.e. $\forall X \forall Y (amazing(X) \leftarrow can_fly(Y))$
if something can fly then everything is amazing!

Variables are *universally quantified within the scope of the clause in which they appear.*

Because all variables appearing in clauses are universally quantified and their scope is the entire clause, quantifiers can be omitted.

Because conditionals can have no conditions, atomic sentences can also contain universally quantified variables.

$likes(bob, X)$

Atomic sentences that do not contain such variables are also called *facts*.

Sorts, types and objects

In unsorted logic
implies

$amazing(X) \leftarrow can-fly(X)$
if a rock can fly then the rock is amazing.

implies

$X + Y = Y + X$
if you add two rocks together in any order,
then the result is the same.

In *sorted or typed logics*, variables are assigned *sorts* or *types*.
In unsorted logic, an extra condition can be added to express the *sort* of a variable.

For example:

$amazing(X) \leftarrow can-fly(X) \wedge animal(X)$
 $animal(X) \leftarrow person(X)$

In object-oriented computing, the person class *inherits* flying from the animal class.

In natural languages, unsorted variables are expressed by words like *anything* and *everything* and sorted variables by common nouns, like *an animal*, *a station*, or *a bird*.

Instead of:

$\forall X (amazing(X) \leftarrow can-fly(X) \wedge animal(X))$

or

$amazing(X) \leftarrow can-fly(X) \wedge isa(X, animal)$
 $isa(X, animal) \leftarrow isa(X, person)$

we write:
or

if an animal can fly then the animal is amazing
any animal that can fly is amazing

Recursive definitions

Conditionals can be used to define *recursive* predicates. The ability to express recursion makes logic programming a general-purpose programming language.

$$\begin{aligned} & \textit{natural-number}(0) \\ & \textit{natural-number}(s(X)) \leftarrow \textit{natural-number}(X) \end{aligned}$$
$$\begin{aligned} & +(0, Y, Y) \\ & +(s(X), Y, s(Z)) \leftarrow +(X, Y, Z) \end{aligned}$$

Notice the treatment and lack of treatment of sorts.

In **functional notation**, the definition is simpler :

$$\begin{aligned} 0 + Y &= Y \\ s(X) + Y &= s(X + Y) \end{aligned}$$

or $(X + 1) + Y = (X + Y) + 1$

Goal clauses

An achievement goal is represented by a *goal clause*, which is an existentially quantified conjunction of atoms and negations of atoms:

$$\exists X_1 \dots \exists X_m (C_1 \wedge \dots \wedge C_n \wedge \neg D_1 \wedge \dots \wedge \neg D_m)$$

If m is 0, then the *goal clause* is called a *definite goal clause*.

Because all variables in goal clauses are existentially quantified it is normal to omit the quantifiers. For example:

stands for

$$\text{likes}(\text{bob}, X)$$
$$\exists X \text{ likes}(\text{bob}, X)$$

Definite clauses and definite goal clauses are also called *Horn clauses* after the logician Alfred Horn. Horn clauses are equivalent to Turing Machines, which are the standard mathematical model of mechanical computation.

In logic programming, *goal clauses* represent the computation to be performed:

$$+(s(s(0)), s(s(0)), X) \wedge +(X, Y, s(s(s(s(s(0)))))))$$

represents the problem of solving the equations $2+2 = X$ and $X+Y = 5$.

Ambiguity = when an expression can be translated into precise language in more than one way

$$X + Y = Y + X$$

$$X + 1 = 2$$

likes(bob, X)

Ambiguity \neq vagueness

Vagueness = when an expression does not have a precise meaning

strong(superman)

big(12334)

small(3)

Logical equivalences

is equivalent to: $\forall X \forall Y (amazing(X) \leftarrow can-fly(Y))$
 $\forall X (amazing(X) \leftarrow \exists Y can-fly(Y))$

are equivalent to: $amazing(X) \leftarrow can-fly(X)$
 $amazing(X) \leftarrow movie-star(X)$
 $amazing(X) \leftarrow (can-fly(X) \vee movie-star(X))$

is equivalent to: $generous-to(X, Z) \leftarrow likes(X, Y) \wedge gives(X, Y, Z)$
 $(generous-to(X, Z) \leftarrow likes(X, Y)) \leftarrow gives(X, Y, Z)$

The symbol \vee is used for the logical connective *or*.
Expressions connected by \vee are called *disjunctions*.
In general, a *disjunction* has the form:

i.e. $C_1 \vee \dots \vee C_n$
 C_1 or ... or C_n

Maintenance goals:

$hungry(me) \rightarrow \exists X eat(me, X)$

$attacks(X, me) \rightarrow runaway(me) \vee attacks(me, X)$

Existential quantifiers in the conclusions of conditional goals are so common, that it is convenient to omit them:

Variables in the conclusion of a conditional goal that are not in the conditions are existentially quantified, with scope the conclusion of the goal.

For example:

Maintenance goal: $hungry(me) \rightarrow eat(me, X)$

The inclusion of disjunctions in the conclusions of conditionals gives the logic of conditionals the power of *classical logic*.

Tip: Existential quantifiers are generalised disjunctions.
Universally quantifiers are generalised conjunctions.

Negation

In classical logic, negative and positive sentences have the same status.
But in Computational Logic, positive sentences are more basic than negative sentences.

Conditionals normally have only positive conclusions,
but may have negative conditions $\neg C$ (also written *not C*).

$$\begin{aligned} \text{liable-to-penalty}(X) &\leftarrow \text{press-alarm}(X) \wedge \text{not emergency} \\ \text{can-fly}(X) &\leftarrow \text{bird}(X) \wedge \text{not penguin}(X) \end{aligned}$$

not C holds if *C* fails to hold. This is called *negation as failure*.

If we are told *bird(john)*, but have no reason to believe *penguin(john)*,
it follows by negation as failure that *can-fly(john)*.

Here is a definition of the odd and even numbers:

$$\begin{aligned} \text{even}(0) \\ \text{even}(s(s(X))) &\leftarrow \text{even}(X) \\ \text{odd}(X) &\leftarrow \text{not even}(X) \end{aligned}$$

Because it cannot be shown that *even(s(0))*, it follows that *odd(s(0))*.

Constraints = maintenance goals with conclusion *false*.

For example, in the context of an agent monitoring its candidate actions, the constraint:

i.e. $liable\text{-to-penalty}(X) \rightarrow false$
Do not be liable to a penalty.

functions as a prohibition,
which prevents actions that are liable to a penalty.
like pressing the alarm signal button improperly or
failing to pay taxes.

Other constraints:

i.e. $even(X) \wedge odd(X) \rightarrow false$
Nothing is both odd and even

Functions, relations and equality

We use function symbols to construct composite terms. Other kinds of functions are treated as relations (or predicates), as in relational databases. Instead of $f(X) = Y$, where f is a function symbol, we write $f(X, Y)$, where f is a predicate symbol. The fact that the relation is a function is represented by the constraint:

$$f(X, Y_1) \wedge f(X, Y_2) \rightarrow Y_1 = Y_2$$

We combine this relational representation of functions with a simple notion of equality, understood as identity, and defined by

$$X = X$$

This representation works well only if individuals have unique names. For example, it's not good enough to say *bob stops the train* if same person is also called *robert* and if more than one person is also called *bob*. We have to give *bob* a unique name, *007* for example, and say something like:

stops(007, the train)
first-name(007, bob)
first-name(007, robert)
first-name(008, bob)

The definition of equality as identity, means that two individuals are identical if and only if they have the same unique name. This contrasts with the more conventional notion of equality, in which the same individual can have several names. For example:

the morning star = the evening star
doctor jekyll = mister hyde

Classical Logic

The syntax of classical logic is an extension of the syntax of the conditional logic in this course. Sentences can be constructed using arbitrary combinations of \rightarrow , \wedge , \vee , \neg , and \forall and \exists .

In conditional logic, all quantifiers can be omitted. But in classical logic, all quantifiers need to be explicit

In conditional logic, there is only way to express that *all birds can fly* and *John is a bird*:

$can\text{-}fly(X) \leftarrow bird(X)$
 $bird(john)$

But in classical logic, the same sentences can be expressed in many logically equivalent ways, including:

$\neg(\exists X((\neg can\text{-}fly(X) \wedge bird(X)) \vee \neg bird(john)))$
 $\neg(\exists X((\neg can\text{-}fly(X) \vee \neg bird(john)) \wedge (bird(X) \vee \neg bird(john))))$

To translate classical logic into conditional logic, it is necessary to use such equivalence-preserving rules of inference as:

replace $\neg \exists X \neg A$ by $\forall X A$
replace $\neg A \vee \neg B$ by $\neg(A \wedge B)$
replace $A \vee \neg B$ by $A \leftarrow B$.

In conditional logic, existential quantifiers are avoided by giving everything that exists a name. Instead of $\exists X bird(X)$, we say $bird(john)$ or $bird(007)$.

If you know that *john is a bird*, why conceal John's identity by saying only that *someone is a bird*.

The relationship among classical logic, clausal logic and Computational Logic

Any sentence of classical logic can be translated into a set of *clauses* of the form:

$$C_1 \wedge \dots \wedge C_n \rightarrow D_1 \vee \dots \vee D_m$$

where each condition C_i and conclusion D_j is an atomic formula, and all variables in the clause are implicitly universally quantified with scope the entire clause.

If n is 0, then $C_1 \wedge \dots \wedge C_n$ is equivalent to *true*.

If m is 0, then $D_1 \vee \dots \vee D_m$ is equivalent to *false*.

Traditionally, such clauses are written in the logically equivalent form:

$$\neg C_1 \vee \dots \vee \neg C_n \vee D_1 \vee \dots \vee D_m$$

Although classical logic can be translated into clausal form, the original sentence and its translation are not always logically equivalent.

For example, the sentence $\forall X \exists Y (mother(X, Y) \leftarrow person(X))$ can be translated into the clause $mother(X, mom(X)) \leftarrow person(X)$.

The clause uses a function symbol, and is more informative than the original sentence.

Composite events + rules + composite actions

$\forall A T_1 T_2 T$ [*pre-sensor detects possible fire in area A at time T_1 \wedge
smoke detector detects smoke in area A at time T_2 \wedge
 $|T_1 - T_2| \leq 60 \text{ sec} \wedge \max(T_1, T_2, T)$*

$\rightarrow \exists T_3 T_3' T_4 T_5$ [*activate local fire suppression in area A at time T_3 \wedge
 $T < T_3 \leq T + 10 \text{ sec} \wedge$*

\rightarrow *fire in area A at time T_4*

send security guard to area A at time T_5

$\wedge T_3 < T_4 \leq T_3 + 30 \text{ sec} \wedge$

$\wedge T_4 < T_5 \leq T_4 + 10 \text{ sec}$

\vee *call fire department to area A at time T_3' $\wedge T < T_3' \leq T + 120 \text{ sec}$]*

Example – simplified conversation

Maintenance goal:

$\forall T_1 T_2 [\text{sentence}(\text{you}, T_1, T_2) \rightarrow \exists T_3 T_4 [\text{sentence}(\text{me}, T_3, T_4) \wedge T_2 < T_3 < T_2 + 3 \text{ sec}]]$

Beliefs:

$\text{adjective}(\text{Agent}, T_1, T_2) \leftarrow \text{word}(\text{Agent}, \text{my}, T_1, T_2)$
 $\text{adjective}(\text{Agent}, T_1, T_2) \leftarrow \text{word}(\text{Agent}, \text{your}, T_1, T_2)$

$\text{noun}(\text{Agent}, T_1, T_2) \leftarrow \text{word}(\text{Agent}, \text{name}, T_1, T_2)$
 $\text{verb}(\text{Agent}, T_1, T_2) \leftarrow \text{word}(\text{Agent}, \text{is}, T_1, T_2)$
 $\text{noun}(\text{Agent}, T_1, T_2) \leftarrow \text{word}(\text{Agent}, \text{bob}, T_1, T_2)$
 $\text{noun}(\text{Agent}, T_1, T_2) \leftarrow \text{word}(\text{Agent}, \text{what}, T_1, T_2)$

$\text{sentence}(\text{Agent}, T_1, T_3) \leftarrow \text{noun-phrase}(\text{Agent}, T_1, T_2)$
 $\wedge \text{verb-phrase}(\text{Agent}, T_2, T_3)$
 $\text{noun-phrase}(\text{Agent}, T_1, T_3) \leftarrow \text{adjective}(\text{Agent}, T_1, T_2)$
 $\wedge \text{noun}(\text{Agent}, T_2, T_3)$
 $\text{noun-phrase}(\text{Agent}, T_1, T_2) \leftarrow \text{noun}(\text{Agent}, T_1, T_2)$
 $\text{verb-phrase}(\text{Agent}, T_1, T_3) \leftarrow \text{verb}(\text{Agent}, T_1, T_2)$
 $\wedge \text{noun-phrase}(\text{Agent}, T_2, T_3)$
 $\text{verb-phrase}(\text{Agent}, T_1, T_2) \leftarrow \text{verb}(\text{Agent}, T_1, T_2)$

Example – simplified conversation

Observations:

word(you, what, 1, 2)

word(you, your, 3, 4)

word(you, is, 2, 3)

word(you, name, 4, 5)

Actions:

word(me, my, 6, 7)

word(me, is, 8, 9)

word(me, name, 7, 8)

word(me, bob, 9, 10)

Blocks world

Maintenance goal:

$request(on(Block, Place), T1) \rightarrow make-on(Block, Place, T2, T3) \wedge T1 < T2$

Facts:

$on(a, b, 0)$
 $on(b, table, 0)$

Beliefs:

$clear(table, T)$
 $clear(Block, T) \leftarrow \neg \exists X on(X, Block, T)$

$make-on(Block, Place, T, T) \leftarrow on(Block, Place, T)$
 $make-on(Block, Place, T1, T3) \leftarrow make-clear(Block, TB1, TB2)$
 $\wedge make-clear(Place, TP1, TP2) \wedge min(TB1, TP1, T1) \wedge max(TB2, TP2, T2)$
 $\wedge move(Block, Place, T3) \wedge T2 < T3$

$make-clear(Place, T, T) \leftarrow clear(Place, T)$
 $make-clear(Place, T1, T3) \leftarrow on(Block, Place, T1)$
 $\wedge make-clear(Block, T1, T2) \wedge move(Block, table, T3) \wedge T2 < T3$

$possible(move(Block, Place), T) \leftarrow clear(Block, T) \wedge clear(Place, T) \wedge Block \neq Place$
 $initiates(move(Block, Place), on(Block, Place), T)$
 $terminates(move(Block, Place), on(Block, Support), T) \leftarrow on(Block, Support, T)$

The Dining Philosophers

There are five philosophers.
To eat a philosopher needs two forks:



The Dining Philosophers

Facts about the initial state of the world
at time 0:

available(fork₀, 0)

available(fork₁, 0)

available(fork₂, 0)

available(fork₃, 0)

available(fork₄, 0)

adjacent(fork₀, philosopher(0), fork₁, 0)

adjacent(fork₁, philosopher(1), fork₂, 0)

adjacent(fork₂, philosopher(2), fork₃, 0)

adjacent(fork₃, philosopher(3), fork₄, 0)

adjacent(fork₄, philosopher(4), fork₀, 0)

The Dining Philosophers

Maintenance goal:

$time\text{-}to\text{-}eat(philosopher(l), T_1)$
 $\rightarrow dine(philosopher(l), T_2, T_3) \wedge T_1 < T_2 \leq T_1 + 24\text{ hours}$

Belief:

$dine(philosopher(l), T_1, T_5)$
 $\leftarrow think(philosopher(l), T_1, T_2) \wedge$
 $pickup\text{-}forks(philosopher(l), T_2, T_3) \wedge$
 $eat(philosopher(l), T_3, T_4) \wedge$
 $putdown\text{-}forks(philosopher(l), T_4, T_5)$

In addition we need to define when actions are possible and what they initiate and terminate.

Homework 2

1. Write a list of all the notation and terminology introduced in this lecture (chapter A1) with a brief explanation. For example:

atom = smallest sentence consisting of a predicate symbol and list of arguments.

\wedge = the symbol meaning “and”.

etc.

2. Let *parent(X, Y)*, *grandparent(X, Y)*, *mother(X, Y)*, *ancestor(X, Y)*, *female(X)* express that *X is a parent of Y*, *X is a grandparent of Y*, *X is mother of Y*, *X is ancestor of Y* and *X is female*, respectively. Write the recursive definitions of *grandparent* and *ancestor*. For example, here is the definition of *mother*:

$$\textit{mother}(X, Y) \leftarrow \textit{parent}(X, Y) \wedge \textit{female}(X)$$

3. Represent the following as a logic program:

Zero is even.

The successor an even number is odd.

The successor of an odd number is even.

Homework 3

From exercise 3, Chapter 1, Logic for
Problem-Solving, on my homepage:

Express in clausal form the assumptions:

Every heavenly creature worth discussing is a star, planet
or comet.

Venus is a heavenly creature, which is not a star.

Comets near the sun have tails.

Venus is near the sun but does not have a tail.

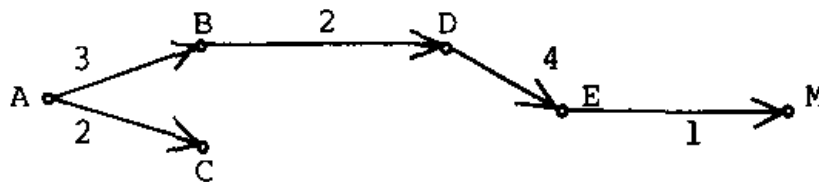
What "obvious" missing assumption needs to be added to the
clauses above for them to imply the conclusion

Venus is a planet ?

Homework 3

From exercise 7, Chapter 1, Logic for Problem-Solving, on my homepage. Note that I have changed the convention for naming constants, variables, predicate symbols and function symbols to the Prolog convention. In the last sentence, $\text{Dist}(x, y, z)$ should be $\text{Distance}(x, y, z)$.

7) Assume that arcs in a directed graph, e.g.



are described by assertions of the form

```
Distance(r,s,t) ←  
  (the length of the arch from r to s is t).
```

Thus the assertion

```
Distance(A,B,3) ←
```

describes the arc from A to B. Assume also that the relationship

```
Plus(x,y,z),
```

which holds when $x+y = z$, is already given. Using only one clause, extend the definition of the relationship $\text{Dist}(x,y,z)$ so that it expresses that there is a path of length z from x to y .

Computational Logic and Human Thinking

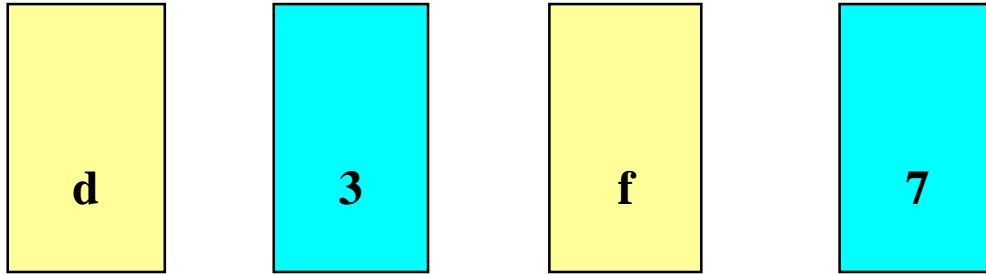
Lecture 3 – The Psychology of Logic – Chapter 2

The Wason selection task

The Byrne Suppression example

The Pollock red light example

The Wason selection task (Peter Wason, 1968)



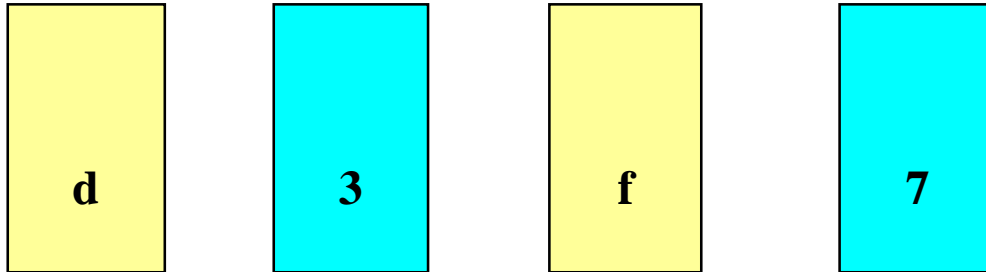
The task is to select those and only those cards that need to be turned over, to determine whether the following conditional holds:

*If there is a d on one side,
then there is a 3 on the other side.*

Only about 10% of the subjects give the answer that is logically correct according to classical logic.

Most psychologists conclude that logic has little relationship with human thinking.

The Wason selection task (Peter Wason, 1968)



*If there is a d on one side,
then there is a 3 on the other side.*

Most people **correctly** turn over the d card,
To check there is a three on the other side.
(**modus ponens**)

Most people **incorrectly** turn over the 3 card,
To check there is a d on the other side.
(the **fallacy** of **affirmation of the consequent**)

Most people **incorrectly** fail to turn over the 7 card,
To check there is no d on the other side.
(**failure** to perform **modus tollens**)

The drinking task

*If a person is drinking alcohol in a bar,
then the person is at least eighteen years old.*

Most subjects give answers that are logically correct according to classical logic:

If a person is drinking beer, they **correctly** check that the person is at least eighteen years old. (**modus ponens**)

If a person is obviously less than eighteen years old, they **correctly** check that the person is not drinking alcohol. (**modus tollens**)

If a person is at least eighteen years old, they **do not incorrectly** check that the person is drinking alcohol. (**affirmation of the consequent**)

The most popular explanation (Leda Cosmides, 1985, 1989) is that humans have **evolved a specialized algorithm** for detecting cheaters in social contracts (in the field of **evolutionary psychology**):

*If you accept a benefit,
then you must meet its requirement.*

Cheng and Holyoak (1985) argue that people reason using **pragmatic reasoning schemes** involving **deontic** notions of permission, obligation and prohibition.

General-purpose versus special-purpose reasoning

Both Cosmides and Cheng and Holyoak argue that people do not have an in-built, general-purpose ability for abstract logical reasoning, but employ specialised algorithms for dealing with practical problems.

An alternative explanation

- Specialised algorithm
= specialised knowledge or beliefs
+ general-purpose reasoning.
- Natural language conditionals
≠ logical conditionals in the LOT.
- Reasoning with conditional beliefs
≠ reasoning with conditional goals.
- Reasoning with negation is problematic,
because “negative observations” need to be derived from
positive observations.

An alternative explanation of the selection task

People treat goal conditionals and belief conditionals differently:

- They treat beliefs as logic programming clauses:

All the conditionals with the same conclusion are the *only* conditionals with that conclusion (which *justifies* the “fallacy” of *affirmation of the consequent*).

Almost all *reasoning is backwards or forwards* (which *inhibits modus tollens*).

- They treat goals as sentences of FOL (first-order logic). (which explains when they reason “correctly” with goals, according to classical logic).

The conditional interpreted as a belief – *modus tollens is hard*

Given the positive observation:

the fourth card has number 7 on the number side.

To perform *modus tollens* with the belief:

*if a card X has letter d on the letter side
then the card X has number 3 on the number side.*

it is necessary first to derive the negative conclusion:

it is not the case that the fourth card has number 3 on the number side.

But this derivation is hard to motivate. Why not also derive:

*it is not the case that the fourth card has number 1 on the number side.
it is not the case that the fourth card has number 2 on the number side.
it is not the case that the fourth card has number 4 on the number side.
....etc.*

The suppression task (Ruth Byrne 1989)

Consider the following pair of premises:

*If she has an essay to write, then she will study late in the library.
She has an essay to write.*

Most people correctly conclude:

She will study late in the library.

Suppose I now say in addition:

If the library is open, then she will study late in the library.

Given this additional information, many people (about 40%) suppress their earlier conclusion that *she will study late in the library*.

Byrne concludes that people do not reason by means of rules of inference (like **modus ponens**), but by constructing **mental models**.

An alternative explanation of the suppression task

The natural language conditional:

if the library is open, then she will study late in the library

is an **incorrect expression** of its intended meaning:

*if the library is **not** open, she will **not** study late in the library.*

In general if A then B is **not equivalent** to if **not** A then **not** B.

The intended meaning of the two sentences is as a **rule and exception**:

*if she has an essay to write, **then** she will study late in the library.
but if the library is **not** open, **then** she will **not** study late in the library.*

But forward and backward reasoning can be viewed as determining **truth in minimal models**.

(which partly justifies the claim about **mental models**)

An alternative representation of the suppression task as a rule with possible exceptions

The natural language conditional:

If she has an essay to write, then she will study late in the library.

has the underlying logical form:

*she will study late in the library
if she has an essay to write
and it is not the case that
she is prevented from studying late in the library.*

*she is prevented from studying late in the library
if the library is not open.*

*she is prevented from studying late in the library
if she is unwell.*

*she is prevented from studying late in the library
if she has a more important meeting.*

*she is prevented from studying late in the library
if she is distracted.*

The use of conditionals to express cause and effect (John Pollock 1995)

Consider the following pair of premises:

an object is red if it looks red.
this apple looks red.

Most people correctly conclude:

this apple is red.

Suppose I now say in addition:

an object looks red if it is illuminated by a red light.

It is likely that you will now withdraw your previous conclusion.

Pollock concludes that people reason by means of sophisticated **argumentation**:
The addition of the second sentence **undermines** the first argument.

An alternative explanation of the looks red example

The natural language conditional:

an object is red if it looks red.

is an incorrect expression of its intended meaning:

an object looks red if it is red.

In general if A then B is **not equivalent** to if B then A.

The intended meaning of the two sentences is to express **cause and effect**:

an object looks red if it is red.

an object looks red if it is illuminated by a red light.

The natural way to express **A causes B**

is:
or equivalently **B if A**
if A then B.

An abductive interpretation of the looks red example

Given: *an object looks red if it is red.*
 an object looks red if it is illuminated by a red light.
 this object looks red.

There are two alternative abductive explanations:

this object is red.
this object is illuminated by a red light.

In general, given beliefs: *A if B1*
 A if B2

 A if Bn
and the observation *A*

abduction derives the alternative explanations : *B1 or B2 or... Bn.*

Conclusions

People (including psychologists, philosophers and logicians) confuse:

Natural language conditionals with conditionals in the language of thought.

Goal conditionals with belief conditionals.

A if B with not A if not B.

A if B with B if A.

Deduction with Abduction.

Whether people are logical (a belief) with
whether people should be logical (a goal).

Whether people are logical when they reason consciously with
whether people are logical when they think subconsciously.

Computational Logic and Human Thinking

Lecture 4 — Chapter A2 Truth

Interpretations and models of sentences

The definition of the truth of a sentence in an interpretation

The paradoxes of the semantics of conditionals

Herbrand interpretations and minimal models

Goedel's incompleteness theorem

Truth and consequences

A sentence C is a *logical consequence* of a set of sentences S (or S *logically implies* C) if (and only if) C is *true* whenever S is *true*.

A set of inference rules is *sound* (or *truth-preserving*) if (and only if) whenever it derives a sentence C from a set of sentences S , then C is a *logical consequence* of S .

A set of inference rules is *complete* if (and only if) whenever a sentence C is a *logical consequence* of a set of sentences S , then there exists a derivation, by means of the inference rules, of C from S .

The notion of *truth* applies only to well-formed formulas that are sentences

A *well-formed formula* is an expression constructed from atomic formulas using the logical connectives, \rightarrow , \wedge , \vee and \neg , and the quantifiers \forall and \exists .

A *sentence* is a well-formed formula all of whose variables are explicitly or implicitly quantified using \forall or \exists .

Truth is *relative* to an interpretation of the symbols of the language.

An *interpretation* is a collection of *individuals* (called the *domain of discourse*), which are the *denotations* (or *meanings*) of the ground terms of the language, together with *relations*, which are the *denotations* of the predicate symbols.

The relations in an interpretation determine the truth values of the atomic sentences of the language, and the truth values of the atomic sentences determine the truth values of all other sentences.

For example

If the constant *john* denotes my cat, the predicate symbols *amazing* and *can-fly* denote being lazy and sleeping all day respectively, then:

$$\textit{amazing}(\textit{john}) \leftarrow \textit{can-fly}(\textit{john})$$

means:

My cat is lazy if my cat sleeps all day.

And because my cat sleeps all day and my cat is lazy,
The atomic sentences *can-fly(john)* and *amazing(john)* are both *true*.

As a consequence, *amazing(john) ← can-fly(john)* is also *true*.

Truth in an interpretation

An *atomic sentence* $p(c_1, \dots, c_n)$ is *true* in an interpretation if (and only if) the individuals denoted by c_1, \dots, c_n are in the relation denoted by p .

A *conjunction* $C_1 \wedge \dots \wedge C_n$ is *true* in an interpretation if (and only if) all of C_i are *true*. (Therefore, if $n = 0$, then the conjunction is *true*.)

A *disjunction* $C_1 \vee \dots \vee C_n$ is *true* in an interpretation if (and only if) at least one of C_i is *true*. (Therefore, if $n = 0$, then the disjunction is not *true*.)

A *conditional* $C \rightarrow D$ is *true* in an interpretation if (and only if) C is *false* or D is *true*.
(Therefore $C \rightarrow \text{false}$ is *true* if and only if C is *false*.)

$\forall X C$ is *true* if (and only if) every *ground instance* of C (obtained by replacing every occurrence of the variable X in C by a ground term) is *true*.

$\exists X C$ is *true* if (and only if) some ground instance of C is *true*.

$\neg C$ is *true* if and only if C is not *true*

An interpretation is a *model* of S
if (and only if) every sentence in S is *true* in the interpretation.

The semantics of conditionals

A conditional (also called *material implication*) of the form $C \rightarrow D$ is logically equivalent to a disjunction $\neg C \vee D$.

This implies that the conditional is *true* whenever D is *true*, no matter whether C is *true* or *false*.

The conditional is also *true* whenever C is *false*, no matter whether D is *true* or *false*. For example, the conditionals:

john can fly \rightarrow $2 + 2 = 4$

the moon is made from green cheese \rightarrow *john can fly*

are both *true* in any interpretation in which $2 + 2 = 4$ is *true* and *the moon is made from green cheese* is *false*, no matter whether *john can fly* is *true* or *false*.

These properties of the semantics of conditionals are known as the *paradoxes of material implication*.

In defence of the semantics of conditionals

john can fly \rightarrow *I am a monkey's uncle*

Assuming the conditional is *true* and that *I am an monkey's uncle* is *false*, it follows that *john can fly* is *false*.

The paradoxes of conditionals can be avoided, partly by invoking pragmatic, rather than semantic, considerations, as argued for example by (Grice, 1989).

For example, why assert the *weak disjunction*, even if it is *true*:

I am going to the party \vee *I will stay at home*

if I have no intention of going to the party,
but I am planning to stay at home instead?

Universal quantifiers

A sentence of the form $\forall X C$ is *true* if and only if every *ground instance* of C is *true*. This definition (called the *substitution interpretation of quantifiers*) works well only if there are enough ground terms in the language to name all the individuals in the interpretation.

The set of ground terms needs to include not only the names of all the individuals in the sentences under consideration, but also additional names for any individuals that might need talking about in the future.

Herbrand interpretations

The definition of truth for arbitrary reduces to the definition of truth for ground atomic sentences.

This means that for many purposes we can ignore what are the real individuals and the real relations denoted by a language, and focus instead on *Herbrand interpretations or Herbrand models*:

A Herbrand interpretation is simply a set of ground atomic sentences, representing the set of all ground atomic sentences true in the interpretation.

Theorem: If a set of sentences has a model, then it has a Herbrand model.

Herbrand interpretations are named in honour of the logician Jacques Herbrand.

Minimal models of definite clause programs

In classical logic, a sentence C is a *logical consequence* of a set of sentences S if (and only if) C is *true* in every model of S .

Typically, S has many, often infinitely many, models.

However, in the case of definite clauses, there is a single model that stands out. It is the Herbrand model M that is generated by instantiating universally quantified variables with ground terms and by reasoning forwards.

E :
 $even(0)$
 $even(s(s(X))) \leftarrow even(X)$

Forward reasoning generates the infinite sequence of atomic sentences:

$even(0), even(s(s(0))), even(s(s(s(s(0))))), \dots$

This set is a Herbrand model of E .

In fact, it is the smallest Herbrand model that makes E *true*.

Minimal models of definite clause programs

The smallest model of a definite clause program H always exists, and it is called the *minimal model* of H . This model is *minimal* in the sense that it is contained in every other Herbrand model of H .

Theorem:

For every definite clause program H , there exists a unique minimal model M such that for all definite goal clauses G :

G is a logical consequence of H
(i.e. G is *true* in all models of H)

if and only if G is *true* in M .

Truth in arithmetic

The standard model of arithmetic is the minimal model of a definite clause program.

Here is a definite clause representation of addition and multiplication, along with a more conventional representation in terms of functions on the right:

$+ (0, Y, Y).$	i.e.	$0 + Y = Y.$
$+ (s(X), Y, s(Z)) \leftarrow + (X, Y, Z).$	i.e.	$s(X) + Y = s(X + Y).$
$\times (0, Y, 0).$	i.e.	$0 \times Y = 0.$
$\times (s(X), Y, V) \leftarrow \times (X, Y, U) \wedge + (U, Y, V).$	i.e.	$s(X) \times Y = (X \times Y) + Y.$

These definite clauses have a unique minimal model, which is the standard model of arithmetic.

The Incompleteness theorem of Kurt Goedel shows that there is no sound and complete set of axioms such that all sentences true in arithmetic are finitely provable from the axioms.

Computational Logic and Human Thinking

Lecture 4 — Chapter A3 Forward and Backward Reasoning

Forward Reasoning

Backward Reasoning

Backward reasoning for computation in logic programming

Soundness and completeness

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

“I have already explained to you that what is out of the common is usually a guide rather than a hindrance. 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.”

“I confess,” said I, “that I do not quite follow you.”

“I hardly expected that you would. Let me see if I can make it clearer. 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.”

Forward Reasoning for definite clauses

Forward reasoning (modus ponens) is more fundamental than backward reasoning, because it is the way that minimal models are generated.

Example: $buys_ticket(john, 150541)$
 $buys_ticket(X, Y) \wedge chosen(Y) \rightarrow rich(X)$

Step 1: Instantiate variables so the fact and one of the conditions of the conditional become identical:

$$buys_ticket(john, 150541) \wedge chosen(150541) \rightarrow rich(john)$$

Step 2: Forward reasoning derives the conclusion. This is classical *modus ponens*:

$$chosen(150541) \rightarrow rich(john)$$

Forward Reasoning with an atomic sentence containing universally quantified variables

likes(bob, X)

likes(X, Y) ∧ gives(X, Y, Z) → generous-to(X, Z)

Step 1: Instantiate variables so the atomic sentence and one of the conditions of the conditional become identical:

likes(bob, X)

likes(bob, X) ∧ gives(bob, X, Z) → generous-to(bob, Z)

Step 2: Forward reasoning derives the conclusion:

gives(bob, X, Z) → generous-to(bob, Z)

Forward Reasoning for definite clauses in general

atomic sentence
conditions \rightarrow *conclusion*

Step 1: Instantiate variables so the atomic sentence and one of the conditions of the conditional become identical:

atomic sentence'
atomic sentence' \wedge *other-conditions'* \rightarrow *conclusion'*.

This should be the most general instantiation that makes the two atoms identical, and is called the (*most general*) *unifier* of the two atoms. All other common instances of the two atoms are instances of this most general unifier. The operation of most general instantiation is called *unification*; and the resulting atoms are said to be *unified*. The unifier of two atoms, if there is one, is *unique* up to the renaming of variables.

Step 2: Delete condition that is now identical to the instantiated atomic sentence:

other-conditions' \rightarrow *conclusion'*.

Note that *atomic sentence'* can occur anywhere in the conditions of the conditional.

Backward reasoning

Initial goal clause: $generous\text{-}to(X, mary)$
Conditional: $likes(X, Y) \wedge gives(X, Y, Z) \rightarrow generous\text{-}to(X, Z)$

Here the variable X in the goal clause is existentially quantified and different from the universally quantified variable X in the conditional, despite having the same (local) name.

Step 1: Instantiate variables so the atomic goal and one of the conclusion of the conditional become identical:

$$generous\text{-}to(X, mary)$$
$$likes(X, Y) \wedge gives(X, Y, mary) \rightarrow generous\text{-}to(X, mary)$$

Step 2: Replace the goal atom by the conditions of the conditional, as subgoals:

Derived goal clause: $likes(X, Y) \wedge gives(X, Y, mary)$

Here the variables X and Y are existentially quantified.

Backward reasoning in general

Initial goal clause:

$$\begin{array}{l} \textit{selected-goal} \wedge \textit{other-goals} \\ \textit{conditions} \rightarrow \textit{conclusion} \end{array}$$

Step 1: Unify the *selected-goal* with the *conclusion* of the conditional.
Apply the unifier to both sentences:

$$\begin{array}{l} \textit{selected-goal}' \wedge \textit{other-goals}' \\ \textit{conditions}' \rightarrow \textit{selected-goal}' \end{array}$$

Step 2: Replace the instantiated selected goal by the conditions of the instantiated conditional:

$$\textit{conditions}' \wedge \textit{other-goals}'.$$

Backward reasoning for computation in logic programming.

Program: $+ (0, Y, Y).$
 $+ (s(X), Y, s(Z)) \leftarrow + (X, Y, Z)$

Initial goal clause: $+ (s(s(0)), s(s(0)), X)$

New goal clause:	$+ (s(0), s(s(0)), X')$	where $X = s(X')$
New goal clause:	$+ (0, s(s(0)), X'')$	where $X' = s(X'')$
New goal clause:	$true$	where $X'' = s(s(0))$

i.e. $X = s(s(s(s(0))))$.

Both forward and backward reasoning are *refutation complete* for Horn clauses.

If G is a definite goal clause and S is a definite clause program, then the following are equivalent:

G is a logical consequence of S .

G is *true* in the minimal model of S .

There exists a derivation of *false*

from the clauses S and $G \rightarrow \textit{false}$

both by forward reasoning and by backward reasoning.

Computational Logic and Human Thinking

Lecture 5 — Search

The relationship between logic and search

The relationship between backward reasoning and and-or trees

The relationship between backward reasoning and or-trees

Breadth-first versus depth-first versus best-first search

Some common misunderstandings about the relationship between logic and search

Paul Thagard (2005) in *Mind: Introduction to Cognitive Science* states on page 45: “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.”

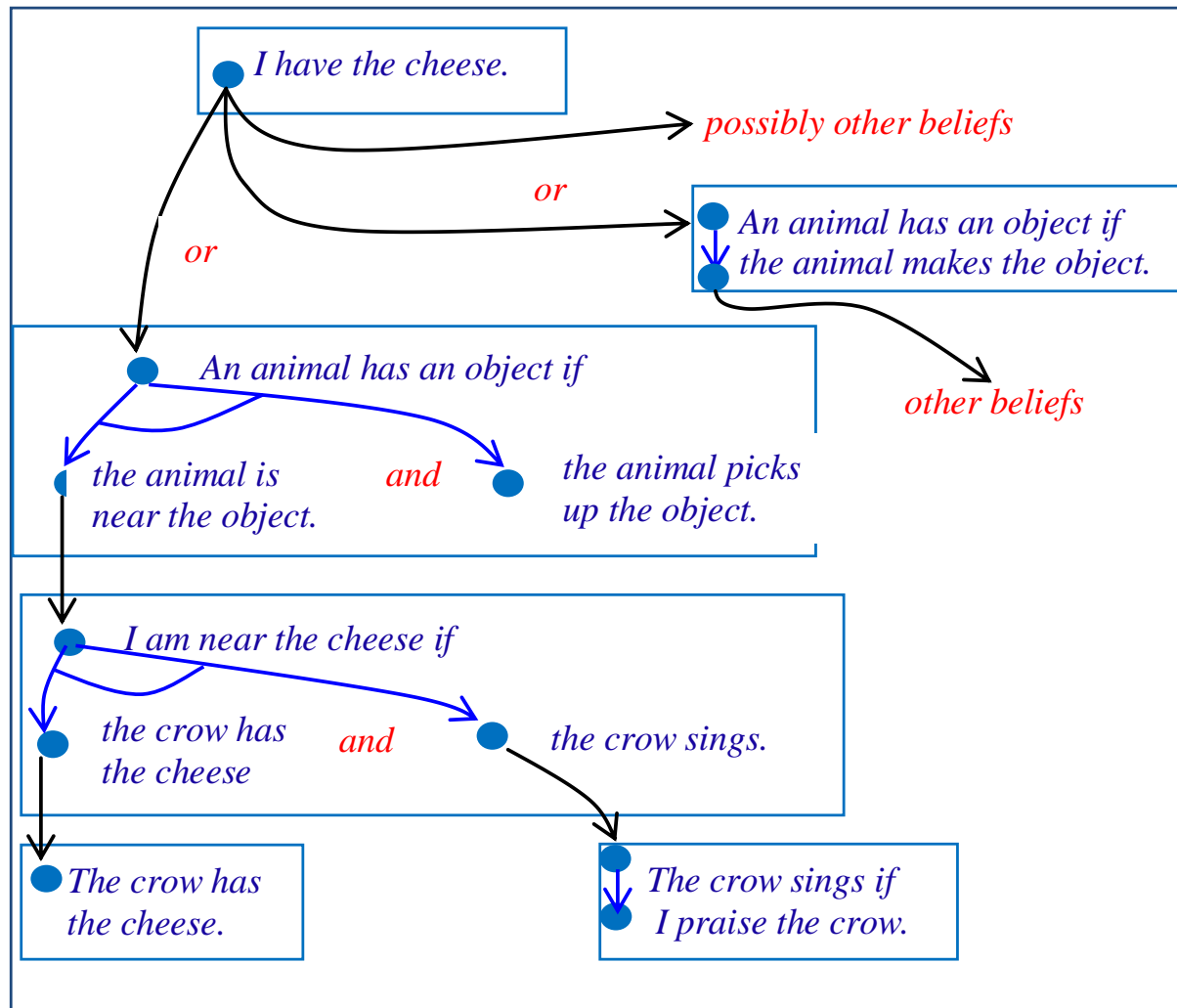
Jonathan Baron (2008) in his textbook *Thinking and Deciding* writes on page 6: “Thinking about actions, beliefs and personal goals can all be described in terms of a common framework, which asserts that thinking consists of *search* and *inference*. We search for certain objects and then make inferences from and about the objects we have found.”

On page 97, Baron states that formal logic is not a complete theory of thinking because it “covers only inference”.

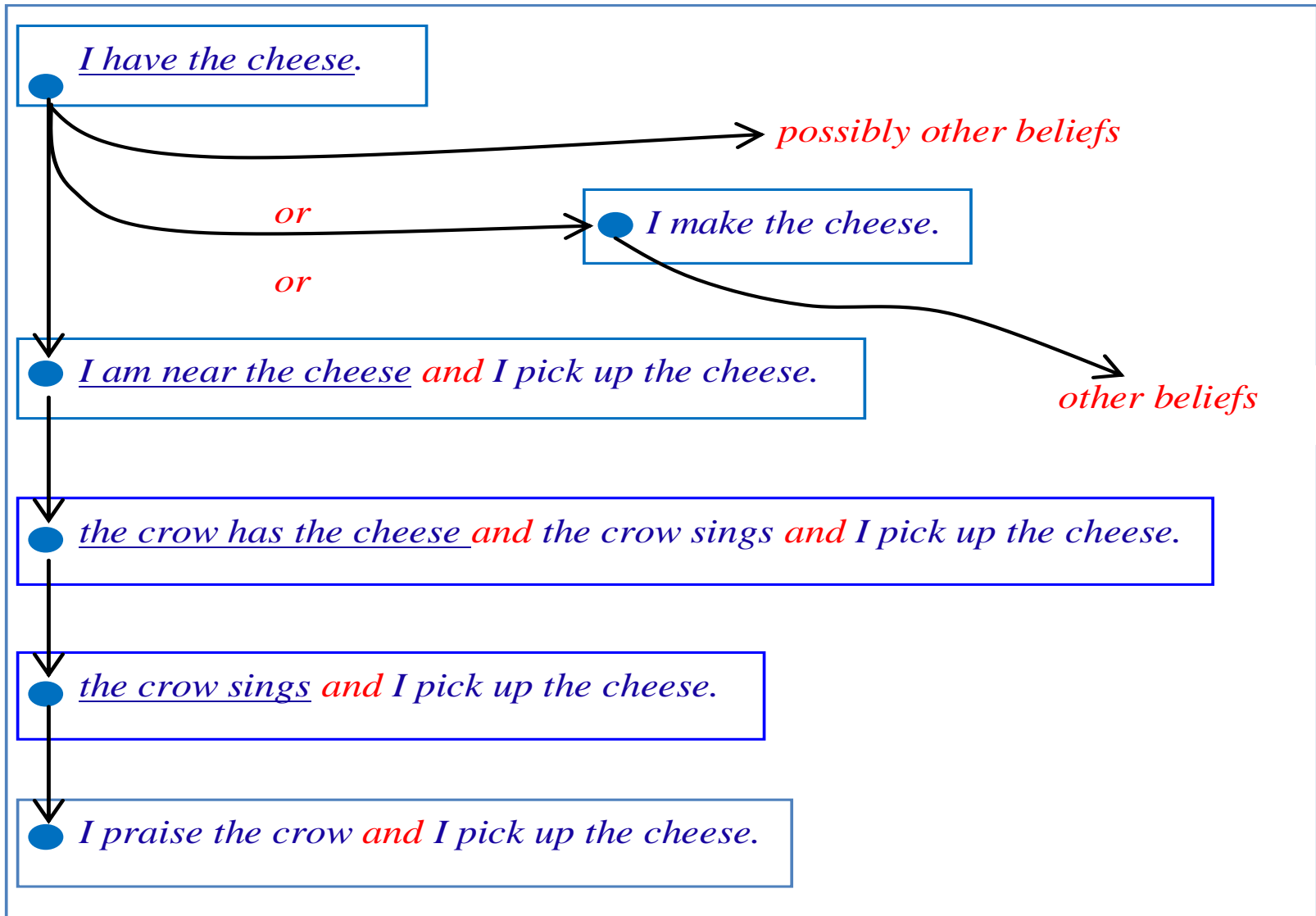
A logical representation of the story of the fox and crow



and-or trees represent the **search space** for backward reasoning



The **search space** for backward reasoning can also be represented as an or-tree



Different strategies can be used for searching the search space

Breadth-first searches level by level, first generating all nodes one step away from the top-level goal, then all nodes two steps away, etc.

If there is any solution to the top-level goal, then breadth-first search is guaranteed to find the shortest solution. But breadth-first search is combinatorially explosive.

If every node has two alternative successor nodes, one level lower in the tree, then if the shortest solution involves two goal-reductions, the search strategy needs to generate only $2^2 = 4$ branches.

If it involves 10 goal reductions, it needs to generate $2^{10} = 1,024$ branches.

But if it needs 50 goal-reductions, then it needs to generate $2^{50} = 1,125,899,906,842,624$ branches.

Depth-first search

Under the same assumptions, if half of the branches contain a solution, at the same level 50 steps away from the top-level goal, then, on the average, depth-first search needs to generate only 100 nodes to find the first solution.

Depth-first search is the opposite of breadth-first search, it explores only one branch at a time, backtracking to try other branches only when necessary. It is very efficient when the search space contains lots of solutions. But it can go disastrously wrong if it contains infinite branches and they are explored before alternative finite branches containing solutions.

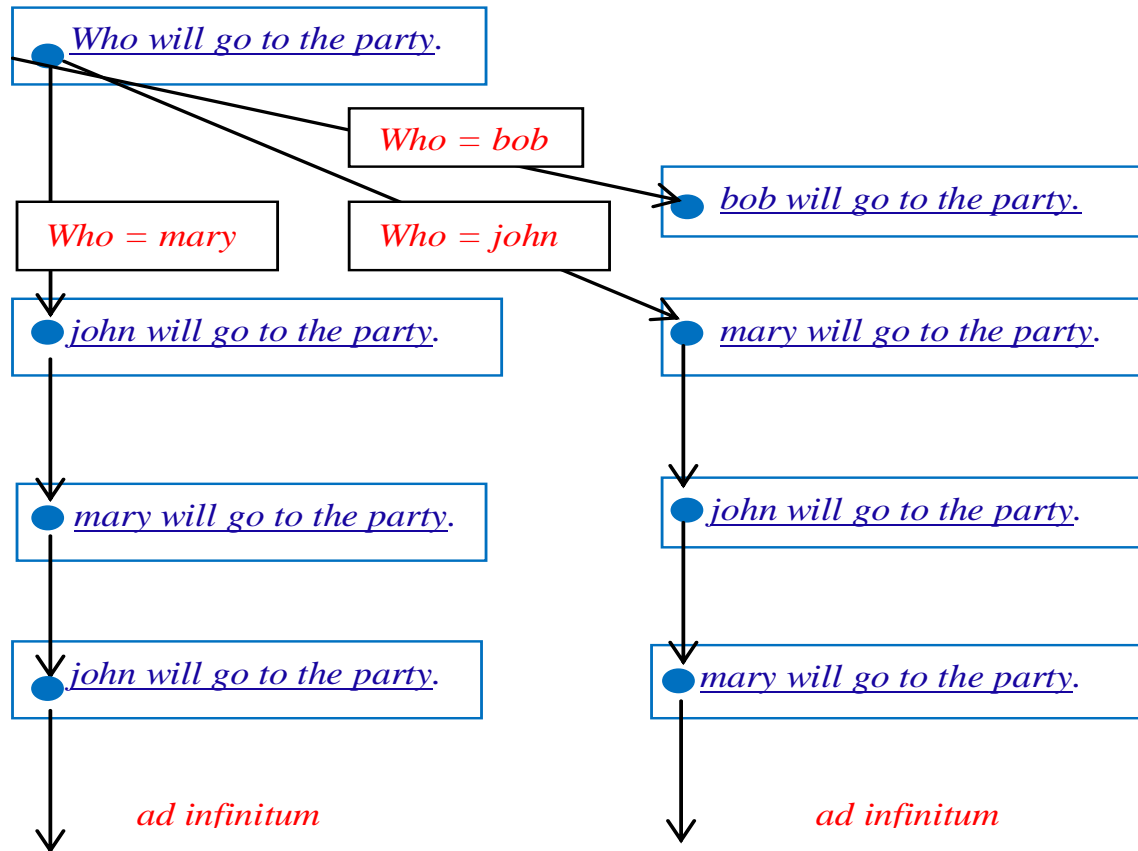
Depth-first search can go into an infinite loop when there are infinitely long branches in the search space

Goal: *Who will go to the party?*

Beliefs: *bob will go to the party.*

mary will go to the party if john will go to the party.

john will go to the party if mary will go to the party.



The programming language Prolog

searches or-trees generated by backward reasoning depth-first.
uses the order in which clauses are written
for the order in which branches are searched.

If the clauses are written in the order:

mary will go to the party if john will go to the party.
john will go to the party if mary will go to the party.
bob will go to the party.

then Prolog goes into an infinite loop.

But if the third sentence *bob will go to the party* is written first, then Prolog finds a solution in one step.

This problem is solved in some implementations of Prolog – for example in XSB Prolog, which stores every subgoal in one place, and recognises and avoids going into most infinite loops.

Best-first search

Best-first search strategies are useful when different solutions of a problem have different values.

For example, if you want to go from A to B, then you might prefer a travel plan that takes the least time, costs the least money or causes the least harm to the environment. No single plan is likely to be best for all of these attributes, so you may have to weigh and trade one attribute off against the other.

Best-first search evaluates partial solutions and explores partial solutions that have the best solution so far in preference to partial solutions that have worse solutions.

Best-first search

Best-first search strategies are useful when different solutions of a problem have different values.

For example, if you want to go from A to B, then you might prefer a travel plan that takes the least time, costs the least money or causes the least harm to the environment. No single plan is likely to be best for all of these attributes, so you may have to weigh and trade one attribute off against the other.

Best-first search evaluates partial solutions and explores partial solutions that have the best solution so far in preference to partial solutions that have worse solutions.

Homework 4 and 5

1. Let S be the following set of clauses:

$\text{on}(a, b)$

$\text{on}(b, c)$

$\text{on}(c, \text{table})$

$\text{on}(d, \text{table})$

$\text{above}(X, Y) \leftarrow \text{on}(X, Y)$

$\text{above}(X, Z) \leftarrow \text{on}(X, Y) \wedge \text{above}(Y, Z)$

- a. Use forward reasoning with S to generate the unique minimal model of S .
- b. Use backward reasoning with S to generate all answers to the goal clause $\text{above}(a, X)$
- c. Demonstrate a Herbrand model of S that is not minimal.

Homework 4 and 5

2. From exercise 3, Chapter 3, page 73. Note the opposite convention for the names of variables, constants and predicate symbols. Note also that top-down is another name for “backward reasoning” and bottom-up is another name for “forward reasoning”. Exercise (7) of Chapter 1 is last week’s homework.

3) Using the clause

```
Distance(x,y,w) ← Distance(x,z,u), Distance(z,y,v), Plus(u,v,w)
```

and any assertions such as

```
Plus(3,2,5) ←  
Plus(5,4,9) ←
```

which are necessary for the Plus relation, construct top-down and bottom-up solutions to the problem

```
← Distance(A,M,w)
```

for the graph shown in exercise (7) of Chapter 1. How many distinct solutions does the top-down search space contain? Is the problem

```
← Distance(x,x,w)
```

solvable?

Computational Logic and Human Thinking

Lecture 6 — Negation as Failure

- Observations are positive.
- Two kinds of negation:
 - Constraints
 - Negation as failure
- Closed world assumption and selective closed world assumption
- Rules and exceptions

Two main sources of negative information

1. We directly observe only positive facts, like:

This card is showing the number 7.

We derive negative facts from a positive fact and an constraint.

This card is not showing the number 3
because no card shows two different numbers at the same time.

2. We directly observe positive facts, like:

There is a train at 10:07

There is a train at 11:07

There is a train at H:07 if $5 < H < 24$

We derive negative facts from the failure to derive positive facts:

There is no train at 10:00

There is no train at 24:00

Other sources of goals and beliefs often do have an essentially negative character in the form of constraints.

Goals and beliefs we have been born with.
Goals and beliefs we have second-hand from
the testimony, persuasion or coercion of other agents.

Nothing is both big and small.
No number is both odd and even.
No letter is both a vowel and a consonant.
Do not drink alcohol in a bar if you are under eighteen years old.
Do not harm a person who is not threatening any harm.
Do not steal.
Do not talk with your mouth full.

Such constraints play an important role in monitoring and eliminating both candidate actions and candidate explanations of observations, as we will see later.

Negative observations can be derived from positive observations, using constraints.

Observation: *the grass is wet.*

Constraint: *if an object is wet
and the object is dry
then false.*

i.e. *it is not the case that
an object is wet and the object is dry.*

Forward reasoning: *it is not the case that the grass is dry.*

Beliefs obtained from experience have a positive bias

Columbus discovered America in 1492.

The last train to leave London for Pulborough is at 22:52

It follows by negation as failure that

Columbus did not discover America in 1493.

Columbus did not discover America in 2012.

The last train to leave London for Pulborough is not at 22:51.

The last train to leave London for Pulborough is not at 22:53.

Programs compute positive, rather than negative facts

Program:

$+(0, Y, Y).$

$+(s(X), Y, s(Z)) \leftarrow +(X, Y, Z)$

computes:

$+(0, 0, 0)$

$+(0, s(0), s(0))$

$+(s(0), 0, s(0))$

.....

No program computes: $\neg +(0, s(0), 0)$

$\neg +(s(0), s(0), s(0))$

$\neg +(s(0), 0, s(0))$

$\neg +(s(0), 0, \textit{pot-of-gold})$

.....

Negation as failure and the closed world assumption

To show that the negation $\neg P$ of a positive sentence holds, show that the positive sentence P does not hold.

The closed world assumption represented as a meta-belief:

For all sentences P ,
 $\neg P$ holds if P does not hold

or $holds(\neg P) \leftarrow \neg holds(P)$

This meta-belief is a meta-sentence, because it talks about sentences.
holds is a meta-predicate.

An note about meta-logic (or modal logic)

Notice the difference between the meta-sentences:

$\text{believes}(\text{john}, \neg \exists X \text{ santa-clause}(X))$
 $\neg \text{believes}(\text{john}, \exists X \text{ santa-clause}(X))$
 $\neg \exists X \text{ believes}(\text{john}, \text{ santa-clause}(X))$

The following sentence is consistent:

$\text{believes}(\text{john}, \exists X \text{ santa-clause}(X)) \wedge \neg \exists X \text{ believes}(\text{john}, \text{ santa-clause}(X))$

The following sentence is consistent:

$\neg \text{believes}(\text{john}, \neg \exists X \text{ santa-clause}(X)) \wedge \neg \text{believes}(\text{john}, \exists X \text{ santa-clause}(X))$

Backward reasoning extended with negation as failure (naf)

Backward reasoning uses the belief:

positive conclusion if positive conditions and negative conditions

as a goal-reduction procedure:

*to show or make the positive conclusion hold,
show or make the positive conditions hold and
show or make the negative conditions fail to hold.*

Beliefs:

*mary will go if john will go.
john will go if bob will **not** go.*

Backwards reasoning:

Goal , show:

mary will go

Subgoal:

john will go.

Subgoal:

*bob will **not** go.*

Naf: *bob will go.*

Failure: *no!*

Success:

yes!

Negation as failure is a form of *defeasible* (= *non-monotonic or default*) reasoning

Old beliefs: *mary will go if john will go.*
 *john will go if bob will **not** go.*
New belief: *bob will go*

Same goal, show: *mary will go*

Subgoal: *john will go.*
Subgoal: *bob will **not** go.*

Naf: *bob will go.*
Success: *yes!*

Failure: *no!*

The new information *bob will go* defeats the previous argument that *mary will go*.

Negation as failure can need an infinite amount of resources

Beliefs: *mary will go if john will go.*
 john will go if mary will go.

Goal, show: *mary will go.*
Subgoal: *john will go.*
Subgoal: *mary will go.*
ad infinitum

Since it cannot be shown that *mary will go*, it follows from the closed world assumption that *mary will not go*. Similarly *john will not go*. *As far as we know*.

In this cases, the infinite chain of reasoning can be detected finitely by noticing that the same subgoal reoccurs as a subgoal of itself.

But in the general case, infinite failure cannot be detected by finite means.

Selective closed world assumption

Robert Moore (1985) gives the following example of a selective closed world assumption:

“Consider my reason for believing that I do not have an older brother. It is surely not that one of my parents once casually remarked, “You know, you don’t have any older brothers”. Nor have I pieced it together by carefully sifting other evidence. I simply believe that if I did have an older brother I would surely know about it, and since I don’t know of any older brothers, I must not have any.”

The general closed world assumption:

For all sentences P $holds(\neg P) \leftarrow \neg holds(P)$

The elective closed world assumption:

For selected sentences p $holds(\neg p) \leftarrow \neg holds(p)$

In Moore’s example, the selected sentence p is “*I have an older brother*”.

Default reasoning without closed world assumptions

Expressions of the form *cannot be shown* do not need to be restricted to expressing closed world or selective closed world assumptions:

bob is accused of robbing the bank.

*a person is innocent of a crime
if the person is accused of the crime
and it cannot be shown that
the person committed the crime.*

*a person committed an act
if another person witnessed the person commit the act.*

If it cannot be shown that bob robbed the bank,
then the following sentence is consistent:

bob robbed the bank, and bob is innocent of robbing the bank.

Rules and exceptions

In everyday language, we commonly express a rule without mentioning possible exceptions:

i.e. *all birds fly.*
an animal can fly if the animal is a bird.

rather than:
an animal can fly if the animal is a bird
and the animal is not a penguin
and the animal is not unfledged
and the animal is not injured.

We commonly correct ourselves with seemingly contradictory statements:

an animal cannot fly if the animal is a penguin
an animal cannot fly if the animal is unfledged
an animal cannot fly if the animal is injured.

Rules and exceptions

We commonly say:

As a general rule: *a conclusion holds if conditions hold.*

Exception: *the conclusion does **not** hold
if other conditions hold.*

What we really mean is:

*a conclusion holds if conditions hold
and other conditions do **not** hold.*

The suppression task example expresses a rule and exception the wrong way around

*she will study late in the library if she has an essay to write.
she will study late in the library if the library is open.*

should be expressed in the **standard form of a rule and exception**:

*she will study late in the library if she has an essay to write.
she will **not** study late in the library if the library is **not** open.*

with the more precise intended meaning:

*she will study late in the library
if she has an essay to write and
she is **not** prevented from studying late in the library.*

*she is prevented from studying late in the library
if the library is not open.*

Missing conditions are sometimes used to avoid details

Housing Benefit is a benefit for people on a low income to help them pay their rent. You may be able to get Housing Benefit if you are on other benefits, work part-time or work full-time on a low income.

The word “**may**” indicates that there are other conditions that also need to be satisfied to get Housing Benefit.

a person gets help to pay rent if the person receives housing benefit.

*a person receives housing benefit
if the person is on other benefits
or the person works part-time
or the person works full-time on a low income
and it is not the case that
the person is ineligible to receive housing benefit.*

Hierarchies of rules and exceptions

- Rule 1: All thieves should be punished.
Rule 2: Thieves who are minors should **not** be punished.
Rule 3: Any thief who is violent should be punished.

The intended meaning:

*a person should be punished
if the person is a thief and the person is **not** a minor.*

*a person should be punished
if the person is a thief and the person is a minor
and the person is violent.*

If these are the only conditions under which a person who is a thief should be punished, then it goes without saying that:

*a person **should not** be punished if the person is a thief
and the person is a minor and the person is **not** violent*

A “higher-level” representation

*a person should be punished
if the person is a thief
and it is **not** the case that
the person is an exception to the punishment rule.*

*a person is an exception to the punishment rule
if the person is a minor
and it is **not** the case that
the person is an exception to the exception to the punishment rule.*

*a person is an exception to the exception to the punishment rule
if the person is violent.*

Suppose bob is a thief, and that is all we know about him

Goal/query: *bob should be punished*

Subgoals: *bob is a thief* and
it is not the case that
bob is an exception

Subgoals: *it is not the case that*
bob is an exception

Naf: *bob is an exception*

Subgoals: *bob is a minor* and
it is not the case that
bob is an exception to the exception

Failure: *no!*

Success: *yes!*

Suppose mary is a thief, and a minor,
and that is all we know about her

Goal/query: *mary should be punished*

Subgoal: *mary is a thief* and

it is not the case that mary is an exception

Subgoal: *it is not the case that mary is an exception*

Naf: *mary is an exception*

Subgoal: *mary is a minor* and *it is not the case that
mary is an exception to the exception*

Subgoal: *it is not the case that
mary is an exception to the exception*

Naf: *mary is an exception to the exception*

Subgoal: *mary is violent*

Failure: *no!*

Success: *yes!*

Failure: *no!*

Homework 6

1. Show by backward reasoning and negation as failure what happens to john, who is a thief, a minor, violent and extremely jealous.
2. Show by backward reasoning and negation as failure that $even(s(s(s(0))))$, where

$$\begin{aligned} &even(0) \\ &even(s(X)) \leftarrow \neg even(X) \end{aligned}$$

3. Suppose $p \leftarrow p$. Does the goal $\neg p$ succeed or fail?
4. Suppose $p \leftarrow \neg p$. Does the goal $\neg p$ succeed or fail?

Lecture 7 Chapter 6

How to Become a British Citizen

Homework 3, 4, 5 solutions

The British Nationality Act

The University of Michigan Lease Termination Clause

The World Health and UNICEF Annual Estimates of
National Infant Immunisation Coverage

Homework 4 and 5

2. From exercise 3, Chapter 3, page 73. Note the opposite convention for the names of variables, constants and predicate symbols. Note also that top-down is another name for “backward reasoning” and bottom-up is another name for “forward reasoning”. Exercise (7) of Chapter 1 is last week’s homework.

3) Using the clause

```
Distance(x,y,w) ← Distance(x,z,u), Distance(z,y,v), Plus(u,v,w)
```

and any assertions such as

```
Plus(3,2,5) ←  
Plus(5,4,9) ←
```

which are necessary for the Plus relation, construct top-down and bottom-up solutions to the problem

```
← Distance(A,M,w)
```

for the graph shown in exercise (7) of Chapter 1. How many distinct solutions does the top-down search space contain? Is the problem

```
← Distance(x,x,w)
```

solvable?

Homework 4 and 5

distance(a, b, 3) distance(a, c, 2) distance(b, d, 2) distance(d, e, 4) distance(e, m, 1)
distance(X, Y, W) ← distance(X, Z, U) ∧ distance(Z, Y, V) ∧ plus(U, V, W)

Is the problem distance(X, X, W) solvable?

No. But the attempt to solve the problem by backwards reasoning generates an infinitely large search space.

It is easier to make the search by forward reasoning terminate, generating a finite Herbrand model in the process (ignoring the necessary instances of plus).

{distance(a, b, 3), distance(a, c, 2), distance(b, d, 2), distance(d, e, 4), distance(e, m, 1), distance(a, d, 5), distance(a, e, 9), distance(a, m, 10), distance(b, e, 6), distance(b, m, 7), distance(d, m, 5)}

ELIZABETH II



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—

- (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.

British Nationality Act

Acquisition at Birth

English

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.

Logic Program

X acquires british citizenship by subsection 1.1 at time T
if *X is born in the uk at time T*
and *T is after commencement*
and *Y is father of X* **or**
Y is mother of X
and *Y is a british citizen at time T* **or**
Y is settled in the uk at time T

British Nationality Act

Deprivation of citizenship

English

40.-(2) The Secretary of State may by order deprive a person of a citizenship status if the Secretary of State is satisfied that deprivation is conducive to the public good.

40.-(4) The Secretary of State may not make an order under subsection (2) if he is satisfied that the order would make the person stateless.

Rule and exception

*The Secretary of State may by order deprive a person of a citizenship status **if** the Secretary of State is satisfied that deprivation is conducive to the public good.*

*The Secretary of State may **not** deprive a person of a citizenship status **if** the Secretary of State is satisfied that the order would make the person stateless.*

British Nationality Act

Deprivation of citizenship

Rule and exception

*The Secretary of State may by order deprive a person of a citizenship status **if** the Secretary of State is satisfied that deprivation is conducive to the public good.*

*The Secretary of State may **not** deprive a person of a citizenship status **if** the Secretary of State is satisfied that the order would make the person stateless.*

Logic program combining the rule and exception into one rule

*The Secretary of State may by order deprive a person of a citizenship status **if** the Secretary of State is satisfied that deprivation is conducive to the public good*

***and** the Secretary of State is **not** satisfied that the order would make the person stateless.*

The relationship between rules and exceptions and argumentation

Rule and exception

*The Secretary of State may by order deprive a person of a citizenship status **if** the Secretary of State is satisfied that deprivation is conducive to the public good.*

*The Secretary of State may **not** deprive a person of a citizenship status **if** the Secretary of State is satisfied that the order would make the person stateless.*

Argumentation

An argument based on

*The Secretary of State may by order deprive a person of a citizenship status **if** the Secretary of State is satisfied that deprivation is conducive to the public good.*

is attacked and defeated by an argument based on

*The Secretary of State may **not** deprive a person of a citizenship status **if** the Secretary of State is satisfied that the order would make the person stateless.*

.

British Nationality Act - Acquisition by Abandonment

– another case of rules and exceptions

English

1.-(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.

Logic Programming

*The conclusion of 1.1 holds for a person **if** the person is found newborn abandoned in the uk after commencement **and it is not shown** that the contrary of the conditions of 1.1 hold for the person.*

University of Michigan Lease Termination Clause – Ambiguous use of English language

The University may terminate this lease when the Lessee, having made application and executed this lease in advance of enrolment, is not eligible to enrol or fails to enrol in the University or leaves the University at any time prior to the expiration of this lease, or for violation of any provisions of this lease, or for violation of any University regulations relative to residence or for health reasons, by providing the student with written notice of termination 30 days prior to the effective time of termination,

unless life, limb or property would be jeopardised, the Lessee engages in the sale or purchase of controlled substances in violation of Federal, state, or local law, or the Lessee is no longer enrolled as a student, or the Lessee engages in the use of firearms, explosives, inflammable liquids, fireworks or other dangerous weapons within the building or turns in a false alarm in which case a maximum of 24 hours notice would be sufficient.

University of Michigan Lease Termination Clause – In clear and logical English

The University may terminate this lease by providing the student with written notice of termination 30 days prior to the effective time of termination

- if
 - the Lessee has made application and executed this lease in advance of enrolment and [the Lessee is not eligible to enrol
 - or the Lessee fails to enrol in the University]
 - or the Lessee leaves the University at any time prior to the expiration of this lease
 - or the Lessee violates any provisions of this lease
 - or the Lessee violates University regulations regarding residence
 - or there are health reasons
- and it is not the case that the University may terminate this lease with a maximum of 24 hours notice of termination.

The University may terminate this lease with maximum 24 hours notice

- if life, limb or property would be jeopardised
- or the Lessee engages in the sale or purchase of controlled substances in violation of Federal, state, or local law
- or the Lessee is no longer enrolled as a student
- or the Lessee engages in the use of firearms, explosives, inflammable liquids, fireworks or other dangerous weapons within the building
- or the Lessee turns in a false alarm.

Perhaps the condition “it is not the case that the University may terminate this lease with a maximum of 24 hours notice of termination” was not intended.

Why would the University want to restrict itself?

WUENIC – A Case Study in Rule-based Knowledge Representation and Reasoning (two papers on my webpage)

[Anthony Burton^{1*}, Robert Kowalski², Marta Gacic-Dobo¹, Rouslan Karimov³, David Brown³](#)

Informal rules	(until ~ 2009)
Attempt to formalise the rules	(from ~2007 - 2009)
Formal rules and Prolog implementation	(from ~ 2009)

1 Department of Immunization, Vaccines and Biologicals,
World Health Organization, Geneva, Switzerland

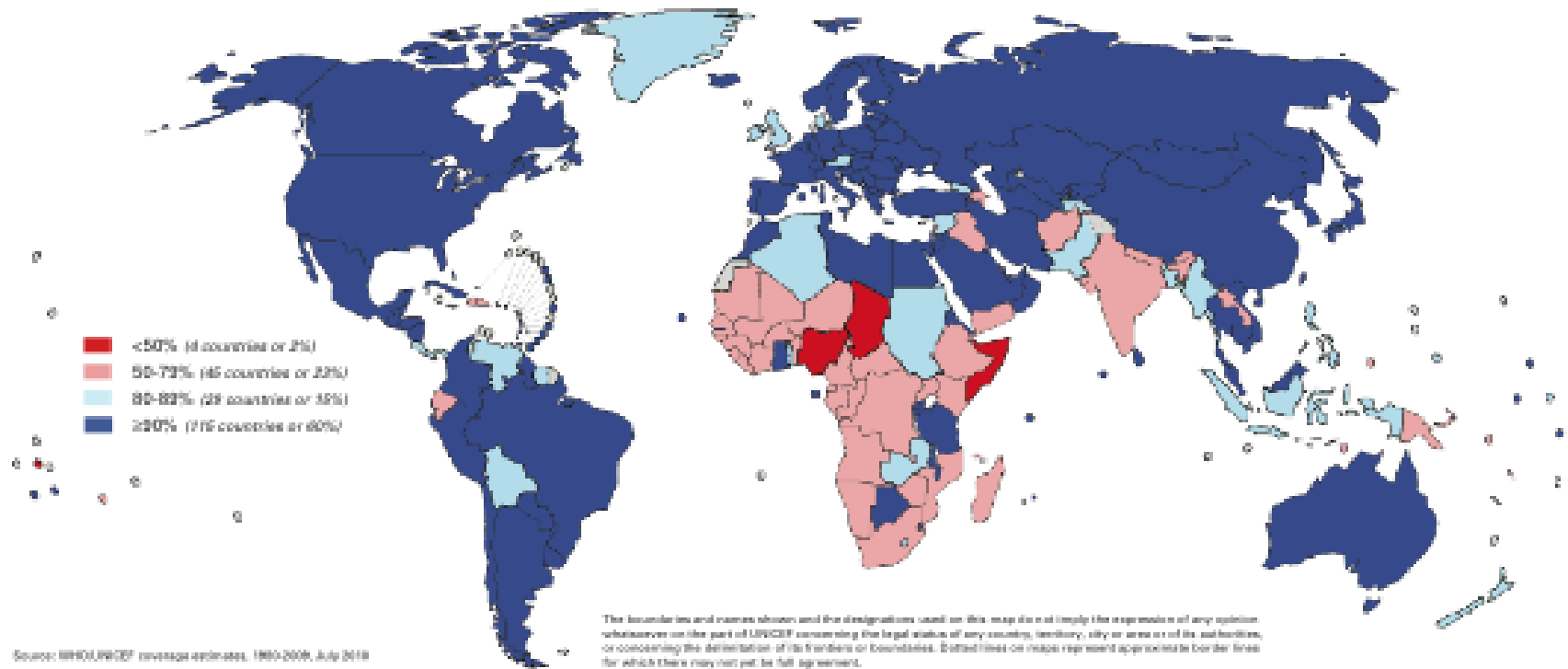
2 Department of Computing, Imperial College London, London, United Kingdom

3 Division of Policy and Practice, United Nations Children's Fund,
New York, New York, United States of America

Yearly Official Infant Immunization Coverage Estimates produced by WHO/UNICEF

144

Map 1: Immunization coverage with measles-containing vaccines in infants, 2009



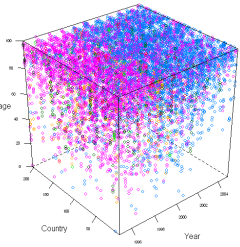
Relationship with Legal Reasoning

No simple, independent way to assess the truth, in this case the real immunisation coverage.

There is a need for a systematic way to make decisions, which are rigorous, transparent, consistent, but also flexible.

Conflicting sources of data

BCG,DTP1,DTP3,Pol3, MCV, HepB3, Hib3, 1980-2006 by countries



Reported data

data points 54497
(27 years, 194 countries)
To UNICEF :17900 records
To WHO: 19481 records
Administrative coverage: 8313 records
Official coverage estimates:8803 records

Survey data

Data points 13017
From 132 countries
cohort coverage 1980 to 2005
DHS survey (194)
MICS survey (211)
EPI cluster survey (147)
Other surveys (185)



Estimates:

24539 data points from 194 countries and 27 years

Nationally reported data =
Numerator/Denominator

Numerator = Number of children immunised

Denominator = Target population of children

Numerator problems:

Missing immunization data

Ad hoc adjustment for missing data

Denominator problems:

Inaccurate population data for denominator

Private / NGO sector not included

Survey data measures percentage directly
(no numerator/denominator problems)

Problems with survey data:

Survey design (e.g., sample size)

Questionnaire design/length

Recall bias (mothers don't remember)

Survey implementation

The rules are represented as logic programs in Prolog

*the WUENIC estimate for the Country, Vaccine and Year is R
if the nationally reported data for a Country, Vaccine and Year is R
and there is no survey data S for the Country, Vaccine and Any-Year
and there is no working group decision to assign an estimate W
for the Country, Vaccine and Year*

*the WUENIC estimate for the Country, Vaccine and Year is W
if there is a working group decision to assign an estimate W
for a Country, Vaccine and Year*

In Prolog: wuenic(Country, Vaccine, Year, R) :-
 reported(Country, Vaccine, Year, R),
 not(survey(Country, Vaccine, Any-Year, S)),
 not(wgd(Country, Vaccine, Year, W)).

 wuenic(Country, Vaccine, Year, W) :-
 wgd(Country, Vaccine, Year, W).

The rules are represented as logic programs in Prolog

*the WUENIC estimate for the Country, Vaccine and Year is **R**
if the nationally reported data for a Country, Vaccine and Year is **R**
and there **is** survey data **S** for the Country, Vaccine and **Year**
and the survey data **S** supports the reported data **R**
and there is no working group decision to assign an estimate **W**
for the Country, Vaccine and Year*

*the WUENIC estimate for the Country, Vaccine and Year is **S**
if the nationally reported data for a Country, Vaccine and Year is **R**
and there **is** survey data **S** for the Country, Vaccine and **Year**
and the survey data **S** does **not** support the reported data **R**
and there is no working group decision to assign an estimate **W**
for the Country, Vaccine and Year*

*the survey data **S** supports the reported data **R**
if the absolute difference between **S** and **R** is **D**
and $D \leq 10\%$*

Prolog code for the estimate between anchor points (two years in which there is a survey), with extra arguments for producing explanations and grade of confidence

```
wuenic(C,V,Y,'C:',Explanation,Coverage,GoC) :-
    reported_time_series(C,V,Y,_,ReportedCoverage,GoCRpt),
    between_anchor_points(C,V,Y,YrBefore,RuleBefore,_,YrAfter,RuleAfter,_),
    not(both_anchors_resolved_to_reported(RuleBefore,RuleAfter)),
    not(workingGroupDecision(C,V,Y,interpolate,_,_,_)),
    calibrate(C,V,YrBefore,YrAfter,Y,Coverage),
    concat_atom(['Reported data calibrated to ',YrBefore,' and ',YrAfter,' levels. '],Explanation),

    goc_calibrated_point(C,V,Y,YrBefore,YrAfter,ReportedCoverage,Coverage,GoCRpt,GoC).

goc_calibrated_point(C,V,Y,YrBefore,YrAfter,ReportedCoverage,Coverage,GoCRpt,GoC) :-
    anchor_point(C,V,YrBefore,_,_,_,GoCBefore),
    anchor_point(C,V,YrAfter,_,_,_,GoCAfter),
    D1 is abs(Y - YrBefore),
    D2 is abs(Y - YrAfter),
    CalFactor is abs(ReportedCoverage - Coverage),
    GoC is ((GoCBefore / D1) + (1 - ((CalFactor / 100) * GoCRpt)) + (GoCAfter / D2)) / 3.
```

The general structure of the WUENIC implementation

Level one. The reported and survey data are evaluated separately, and if necessary are ignored or adjusted.

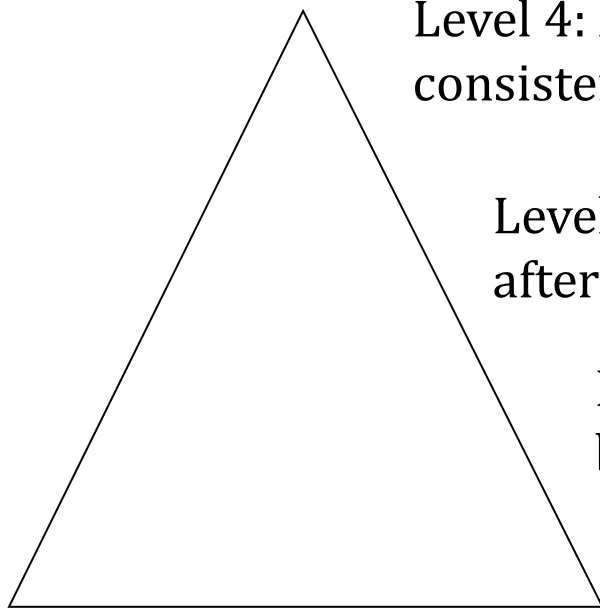
Level two. If data are available from only a single source (national reports or surveys), then the estimates are based on that source alone. Otherwise, the estimates are made at “anchor years”, which are years in which there are both reported data and survey data.

Level three. Estimates are made at non-anchor years, before, between and after anchor years, influenced by the estimates at the anchor years.

Level four. The resulting estimates for the different vaccines are then cross checked for consistency between related vaccines, and adjustments are made if necessary.

Estimates for new years are not based on estimates for previous years.

Forward reasoning fills the triangle bottom-up.
Backward reasoning fills the triangle top-down.



Level 4: Adjust the estimates for consistency among related vaccines.

Level 3: Estimate before, between and after anchor years.

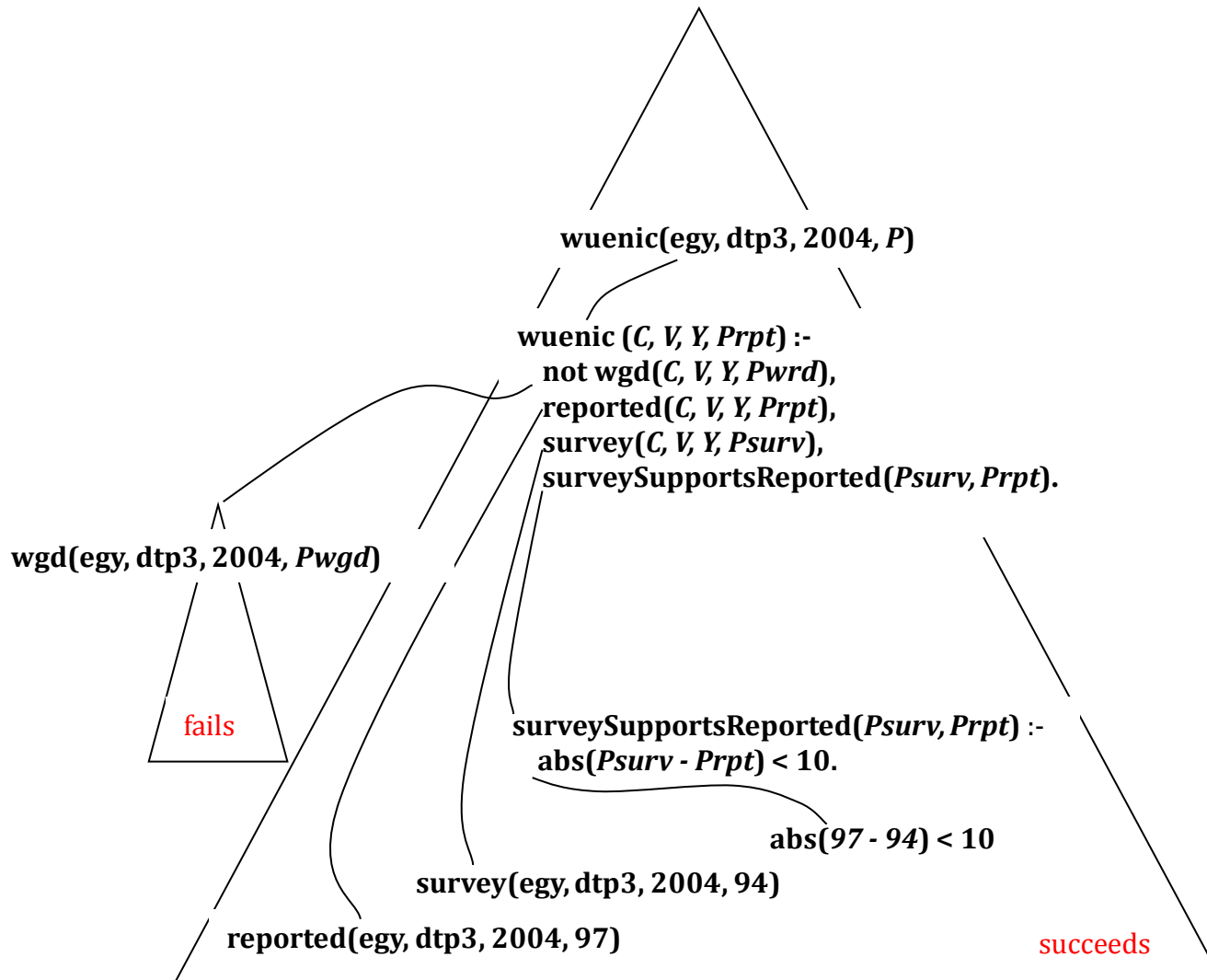
Level 2: Estimate at “anchor years”, with both reported and survey data.

Level 1: “Clean” the data.

Level 0: Unprocessed reported and survey data.

Prolog executes the rules top-down.

The structure of the Prolog program



Problems with the Prolog implementation

Prolog top-down execution repeatedly re-solves lower-level subgoals. 20 minutes per country is too long.

Solution

XSB Prolog top-down execution puts subgoals in a table and solves them only once. 20 seconds per country.

(Tabling was invented in H. Tamaki and T. Sato, OLD resolution with tabulation. Proc. 3rd International Conference on Logic Programming, 1986.)

Homework 7

Consider the set of definite clauses L:

likes(bob, mary)

likes(bob, X) ← likes(X, mary)

1) Which of the following Herbrand interpretations are models of L (i.e. make all clauses in L true)? Which of these is a minimal model of L?

- a. {likes(mary, bob), likes(bob, bob), likes(mary, mary)}
- b. {likes(bob, mary), likes(bob, bob), likes(mary, mary)}
- c. {likes(mary, bob), likes(bob, bob), likes(mary, mary)}
- d. {likes(bob, mary), likes(bob, bob)}

2) In which of the Herbrand interpretations above is the sentence likes(bob, X) → likes(X, mary) (i.e. $\forall X(\text{likes}(\text{bob}, X) \rightarrow \text{likes}(X, \text{mary}))$) true?

Lecture 8 Chapter 7

The Louse and the Mars Explorer

- Thermostat's input-output behaviour
- Fox's input-output behaviour
- Human input-output behaviour on the underground
- Condition-action rules generate behaviour
- The Production system cycle
- Conflict resolution
- Production systems without any representation of the world
- Production systems with memory
- The use of production systems to simulate goal-reduction
- Confusion between production rules and logical implications:

Production systems

Condition-action rules (also called *production rules*) are similar to *descriptions* of behaviour.

However, they are used to *generate* behaviour.

Their conclusions can be expressed in the *imperative* mood:

If conditions then do actions.

rather than in the *declarative* mood:

If conditions then you do actions.

Thermostat's input-output behaviour
described in condition-action terms:

*If current temperature is T degrees
and target temperature is T' degrees
and $T < T' - 2^\circ$
then the thermostat **turns** on the heat.*

*If current temperature is T degrees
and target temperature is T' degrees
and $T > T' + 2^\circ$
then the thermostat **turns** off the heat.*

Thermostat's input-output behaviour
generated by condition-action rules:

*If current temperature is T degrees
and target temperature is T' degrees
and $T < T' - 2^\circ$
then **turn** on the heat.*

*If current temperature is T degrees
and target temperature is T' degrees
and $T > T' + 2^\circ$
then **turn** off the heat.*

Passenger behaviour **described** in
condition-action terms:

*If a passenger observes an emergency on the underground,
then the passenger **presses** the alarm signal button.*

Passenger behaviour **generated** by
condition-action rules:

*If I observe an emergency on the underground,
then I **press** the alarm signal button.*

Fox's behaviour **described** in condition-action terms:

*If the fox sees that the crow has cheese,
then the fox **praises** the crow.*

*If the fox is near the cheese,
then the fox **picks up** the cheese.*

Fox's behaviour **generated** by condition-action rules:

*If I see that the crow has cheese,
then the I **praise** the crow.*

*If I am near the cheese,
then I **pick** up the cheese.*

Production systems

Declarative memory consisting of atomic sentences, and
Procedures consisting of condition-action rules:

If conditions C, then do actions A.

look like logical implications,
but do not have a logical semantics.

Production system cycle:

- read a current input,
- use **forward chaining** to match the input with the conditions **C** of production rules,
- perform **conflict-resolution** to choose a single rule if more than one rule is satisfied, and
- execute the associated actions **A**.

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 with no representation of the world

What it's like to be a louse:

If it's clear ahead, then move forward.

If there's an obstacle ahead, then turn right.

If I am tired, then stop.

Observe: *Clear ahead.*

Do: *Move forward.*

Observe: *Clear ahead.*

Do: *Move forward.*

Observe: *Obstacle ahead.*

Do: *Turn right.*

Observe: *Clear ahead and tired.*

Do: **Conflict**

Production Systems with Memory

What it's like to be a Mars Explorer:

*If the place ahead is clear
and I haven't gone to the place before,
then go to the place.*

*If the place ahead is clear
and I have gone to the place before,
then turn right.*

*If there's an obstacle ahead
and it doesn't show signs of life,
then turn right.*

*If there's an obstacle ahead
and it shows signs of life,
then report it to mission control
and turn right.*

How to remember where you have been

Divide the terrain into squares with co-ordinates (E, N) where E is the distance of the square East, N is its distance North, and (0, 0) is the square where you start.

The world around you:

Life at (2, 1)

Clear at (1, 0)

Clear at (2, 0)

Obstacle at (3, 0)

Obstacle at (2, -1)

Obstacle at (2, 1).

Your behaviour is completely predetermined:

Observe: *Clear at (1, 0)*

Do: *Go to (1, 0)*

Observe: *Clear at (2, 0)*

Do: *Go to (2, 0)*

Observe: *Obstacle at (3, 0)*

Do: *Turn right*

Observe: *Obstacle at (2, -1)*

Do: *Turn right*

Observe: *Clear at (1, 0)*

Remember: *Gone to (1, 0)*

Do: *Turn right*

Observe: *Obstacle at (2, 1) and Life at (2, 1)*

Do: *Report life at (2, 1) to mission control*

Do: *Turn right*

Example (Thagard, page 45)

“unlike logic, rule-based systems can also 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.*

Forward chaining with the rule simulates
backward reasoning with the belief:

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

The use of production systems to simulate goal-reduction

The fox's reduction of the goal of having an object
can be simulated by the condition-action rule:

*If I want to have an object
then add to my beliefs that I want to be near the object
and pick up the object.*

The condition-action rule approach loses
the connection with the belief:

*I have an object
if I am near the object
and I pick up the object.*

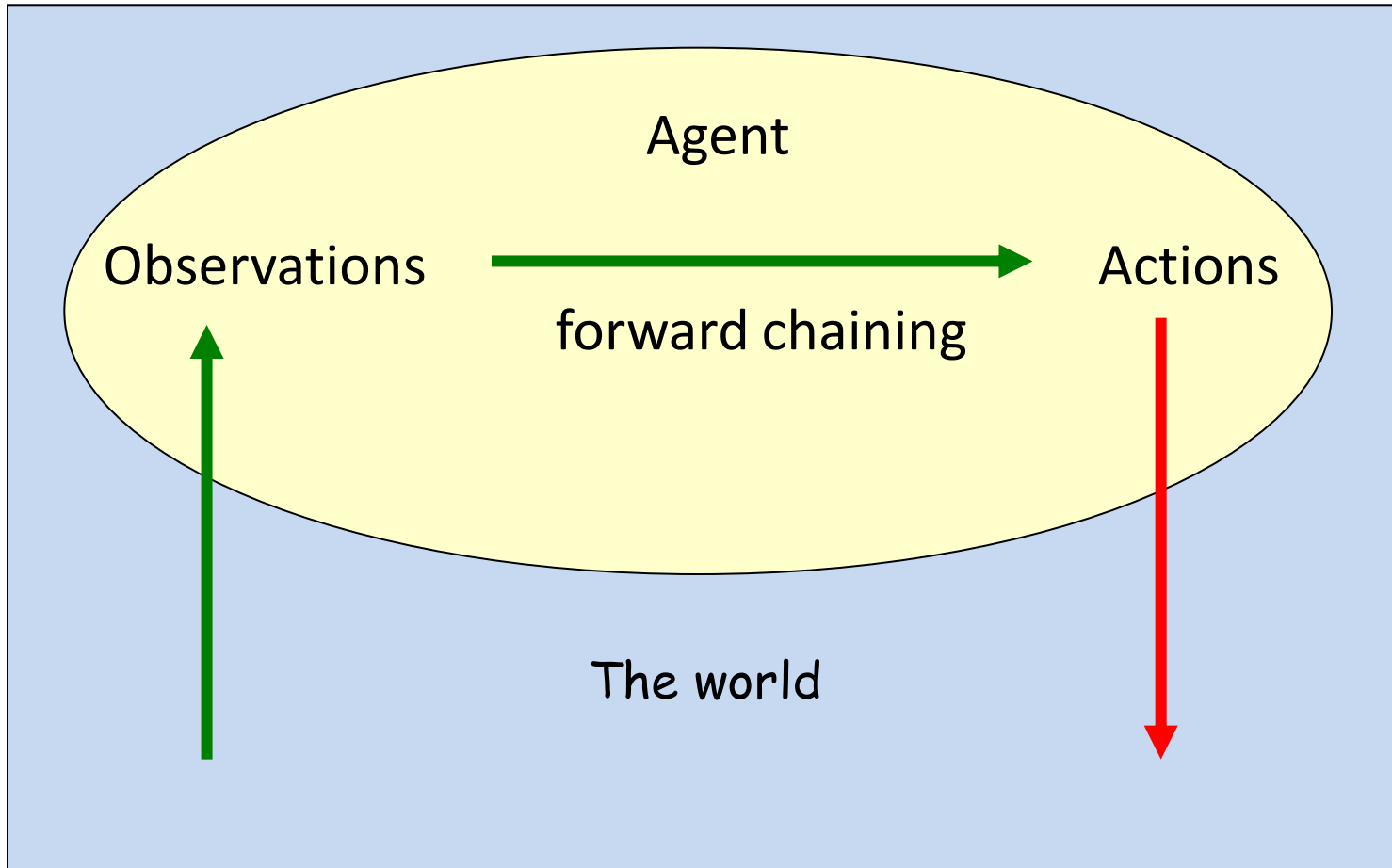
Three kinds of production rules

- **Logical rules** that are used to reason forward (modus ponens).
- **Reactive rules** that implement stimulus-response associations.
- **Pro-active rules** that simulate goal-reduction:

If goal G and conditions C then add H as a sub-goal.

Production rules have an operational,
but not a logical semantics.

The production system view of the relationship between an agent and the world ???



Lecture 9 Chapter 8

Maintenance Goals as the Driving Force of Life

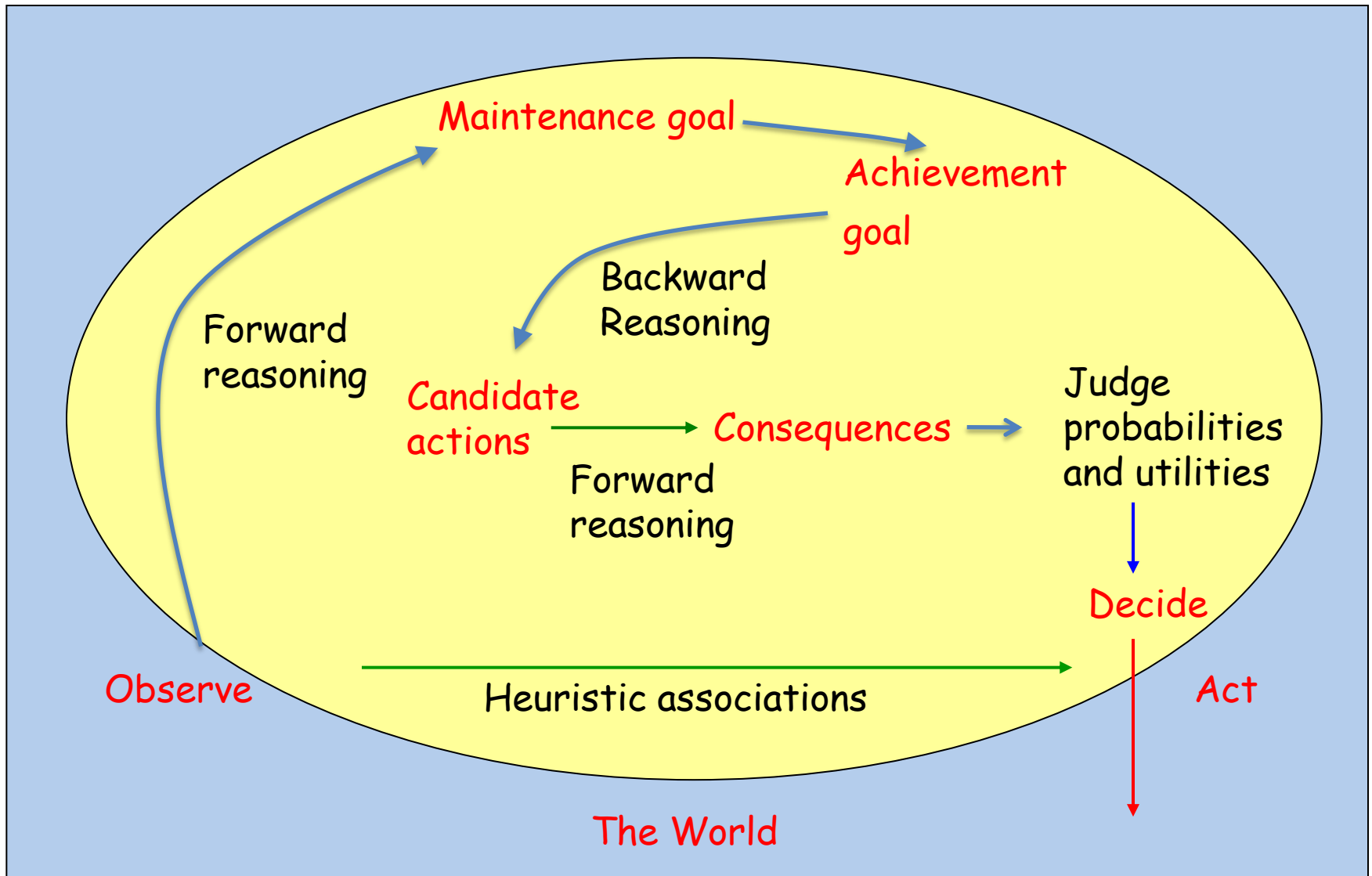
The fox and the crow

The London underground

Prohibitions as maintenance goals

Database integrity constraints as maintenance goals

Maintenance Goals as the Driving Force of Life



Intelligent agent cycle

Cycle to maintain your existence,
observe the world,
think,
decide what actions to perform,
act,
Cycle again to maintain your existence.

Maintenance Goals as the Driving Force of Life

Goal: *if I become hungry, then I have food and I eat the food.*

Beliefs: *an animal has an object
if the animal is near the object
and the animal picks up the object.*

*I am near the cheese
if the crow has the cheese
and the crow sings.*

the crow sings if I praise the crow.

*cheese is a kind of food.
food is a kind of object.*

The fox's agent cycle

Cycle 1

Observation: *I become hungry.*
Forward reasoning
Achievement goal: *I have food and I eat the food.*
No candidate action.

Cycle 2

No observation.
Backward reasoning
New subgoals: *I am near food and I pick up the food and I eat the food.*
No candidate action.

Cycle 3

Observation: *the crow has cheese.*
Forward reasoning
New belief: *I am near the cheese if the crow sings.*
No candidate action.

Cycle 4

No observation.
Backward reasoning
New subgoals: *the crow sings and I pick up the cheese and I eat the cheese.*
No candidate action.

The fox's agent cycle

Cycle 5

No observation.

Backward reasoning

New subgoals: *I praise the crow* and *I pick up the cheese* and *I eat the cheese*.

Action: *I praise the crow*.

Cycle 6

Observation, action succeeded: *I praise the crow*.

Forward reasoning,

Remaining subgoals: *I pick up the cheese* and *I eat the cheese*.

Action: *I pick up the cheese*.

Cycle 7

Negative observation, action failed:

I do not pick up the cheese.

Action, retried:

I pick up the cheese.

Cycle 8

Observation, action succeeded:

I pick up the cheese.

Forward reasoning, remaining subgoal:

I eat the cheese.

Action:

I eat the cheese.

Cycle 9

Observation, action succeeded:

I eat the cheese.

A general pattern of reasoning

- Observation: *An event happens.*
- Forward reasoning: *The event matches a condition of a maintenance goal or belief.*
- Achievement goal: *Eventually, after a combination of forward and backward reasoning, an instance of the conclusion of a maintenance goal is derived as an achievement goal.*
- Backward reasoning: *Beliefs are used to reduce the achievement goal to actions.*
- Actions: *Action subgoals are selected for execution.*
- Observation: *The agent observes whether the actions succeed or fail. Actions that fail are retried if their time limit has not expired.*

The London underground revisited

Maintenance goal: *if there is an emergency then I get help.*

Beliefs: *a person gets help if the person alerts the driver.*

a person alerts the driver

if the person presses the alarm signal button.

there is an emergency if there is a fire.

there is an emergency if one person attacks another.

there is an emergency if someone becomes suddenly ill.

there is an emergency if there is an accident.

A passenger's agent cycle

- Observation: *there is a fire.*
- Forward reasoning, new belief: *there is an emergency.*
- Forward reasoning, achievement goal: *I get help!*
- Backward reasoning, subgoal: *I alert the driver!*
- Backward reasoning, action: *I press the alarm signal button!*

Emergencies on the London underground



Prohibitions can be regarded as maintenance goals whose conclusion is *false*.

if you steal then false.

i.e. *Do not steal.*

*if you are drinking alcohol in a bar
and are under eighteen then false.*

i.e. *Do not drink alcohol in a bar if you are under eighteen.*

*if you are liable to a penalty for performing an action
and you cannot afford the penalty
and you perform the action
then false.*

i.e. *Do not perform an action
if you are liable to a penalty for performing the action
and you cannot afford the penalty.*

Integrity constraints in databases can be regarded as maintenance goals

Update: *Enoch father of Adam*

Database: *Eve mother of Cain*

Eve mother of Abel

Adam father of Cain

Adam father of Abel

Cain father of Enoch

Enoch father of Irad

X ancestor of Y if X mother of Y.

X ancestor of Y if X father of Y.

X ancestor of Z if X ancestor of Y and Y ancestor of Z.

Integrity constraints:

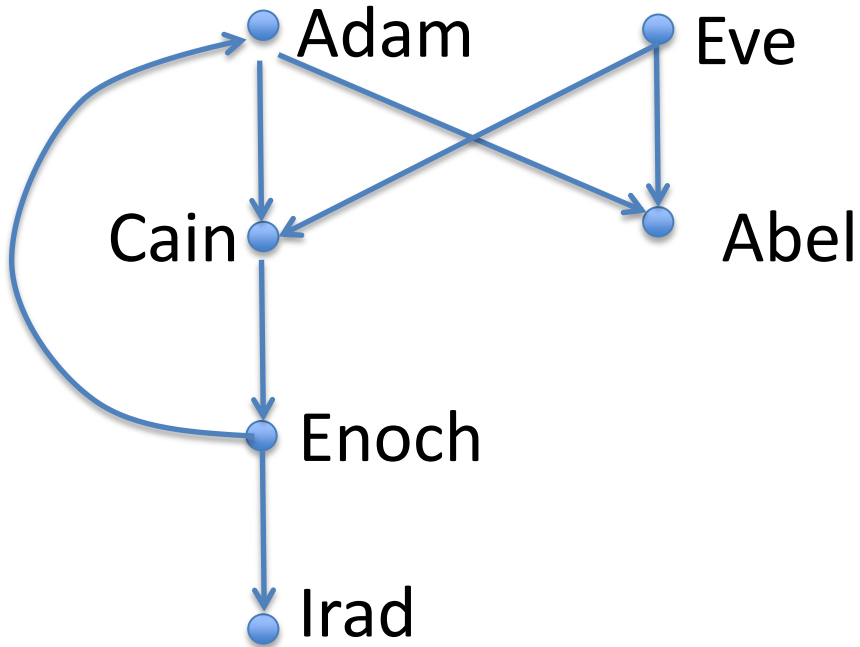
i.e. *if X is the mother of Y and X is the father of Z then false.*

No one is both a mother and a father.

i.e. *if X is an ancestor of X then false.*

No one is their own ancestor.

The ancestor relation represented as a graph



The pattern for assimilating database updates is the same as the pattern for assimilating observations

Update: *Enoch father of Adam*

Forward reasoning: *Enoch ancestor of Adam*

Forward reasoning: *X ancestor of Adam if X ancestor of Enoch*

Backward reasoning: *X ancestor of Adam*
if X ancestor of Y and Y ancestor of Enoch

Backward reasoning: *X ancestor of Adam*
if X ancestor of Y and Y father of Enoch

Backward reasoning: *X ancestor of Adam if X ancestor of Cain*

Backward reasoning: *X ancestor of Adam if X father of Cain*

Backward reasoning: *Adam ancestor of Adam*

Forward reasoning: *false*

Homework 8-9

Here is a production system for operating a mine:

$critical \leq Methane \rightarrow alarm \wedge delete\ operate$
 $critical > Methane \rightarrow add\ operate$
 $operate \wedge high < Water \rightarrow pump$
 $operate \wedge low < Water \wedge pump-active \rightarrow pump$
 $operate \wedge high \geq Water \wedge \neg pump-active \rightarrow nil$

alarm, *pump* and *nil* are actions.

The action *alarm* initiates an alarm, which continues until a human operator turns the alarm off.

The action *pump* initiates *pump-active*, which continues to hold only for a sort time interval of time ϵ and stops after that, unless it is reactivated.

The action *nil* does nothing.

operate can be viewed either as a goal or as a fact.

Homework 8-9

1. Let *critical* = 100, *high* = 20 and *low* = 10. Suppose the methane and water-levels vary as follows. Fill in the missing values for *pump-active*, *alarm* and *pump*

	<i>methane-level</i>	<i>water-level</i>	<i>pump-active</i>	<i>alarm</i>	<i>pump</i>
time ₁	66	18	no		
time ₂	77	20			
time ₃	88	20.0001			
time ₄	99	20.00001			
time ₅	99	15			
time ₆	100	12			
time ₇	110	18			
time ₈	104	19			
time ₉	98	19			
time ₁₀	98	15			

2. Rewrite the production system in logical form as a collection of maintenance goals and beliefs. You may want to add a time argument to the predicates, writing for example *methane-level*(*M*, *T*) , *water-level*(*M*, *T*) , *alarm*(*T*) , *pump*(*T*) , *pump-active*(*T*).

Lecture 10 Chapter A5 The Resolution Rule

Both forward and backward reasoning are special cases of the resolution rule. Resolution also includes such inferences as:

From: *you deal with the emergency appropriately* \leftarrow *you get help.*

you get help \leftarrow *you alert the driver.*

derive: *you deal with the emergency appropriately* \leftarrow *you alert the driver.*

From *true* \rightarrow *playing(bob)* \vee *working(bob)* (i.e. *playing(bob)* \vee *working(bob)*)

playing(bob) \rightarrow *false* (i.e. \neg *playing(bob)*)

derive *working(bob)*

Lecture 10 Chapter A5 The Resolution Rule

In propositional logic (sentences without variables)
given two clauses of the form:

$$\begin{array}{l} D \rightarrow E \vee A \\ A \wedge B \rightarrow C \end{array}$$

where B and D are conjunctions of atoms including the atom *true*, and C and E are disjunctions of atoms including the atom *false*, *resolution* derives the *resolvent*:

$$D \wedge B \rightarrow E \vee C.$$

The two clauses from which the resolvent is derived are called the *parents* of the resolvent, and the atom A is called the atom *resolved upon*.

Forward reasoning is a special case of resolution

Resolution, from:

$$D \rightarrow E \vee A$$

$$A \wedge B \rightarrow C$$

derive:

$$D \wedge B \rightarrow E \vee C.$$

Forward reasoning is the special case where the parents are definite clauses, and one of the parents is a fact, where D is *true* and E is *false*. So $D \rightarrow E \vee A$ is just A

Forward reasoning is a directed form of resolution:

From

A

and


$$A \wedge B \rightarrow C$$

derive

$$B \rightarrow C$$


Backward reasoning is a special case of resolution

Given: $D \rightarrow E \vee A$
 $A \wedge B \rightarrow C$
the *resolvent* is: $D \wedge B \rightarrow E \vee C.$

Backward reasoning is the special case where the parents are definite clauses, and one of the parents is a denial, where C is *false* and E is *false*. So $D \rightarrow E \vee A$ is $D \rightarrow A$ and $A \wedge B \rightarrow C$ is $A \wedge B \rightarrow$

Backward reasoning is a directed form of resolution:

From: $A \wedge B \rightarrow$
and $D \rightarrow A$
derive $D \wedge B \rightarrow$



Two interpretations of backward reasoning.

Interpretation 1: goal reduction

A clause of the form $A \wedge B \rightarrow$
is a goal: show or make $A \wedge B$

Resolution with a clause $D \rightarrow A$
selects one of the subgoals A and reduces it to further subgoals D
remembering the other subgoals B
deriving the new subgoals

$$D \wedge B$$

Interpretation 2: generalised modus tollens

A clause of the form $A \wedge B \rightarrow$
is a denial/constraint $\neg [A \wedge B]$

Resolution with a clause $D \rightarrow A$
to derive $\neg [D \wedge B]$

generalises **modus tollens**:

From $\neg [A]$ and $D \rightarrow A$ derive $\neg [D]$

Modus tollens is the problematic inference in the Wason selection task.

Two interpretations of backward reasoning. The relationship between them

The denial/constraint asserts that the goal has no solution.

$$\neg [A \wedge B]$$
$$A \wedge B$$

To show that the goal does have a solution, resolution attempts to **refute** the assertion, by deriving *false* (a contradiction).

The atomic sentence *false* can also be interpreted as an empty goal (or empty clause).

In general, resolution is used in **refutation procedures**, which show that:

a sentence *C* is a logical consequence of a set of sentences *S* by showing that the set of sentences $S \cup \{\neg C\}$ is inconsistent, by deriving the empty clause *false*.

For example, to show *A* is a logical consequence of *A*, show $\{A, \neg A\}$ is inconsistent. Resolution derives the empty clause.

This is equivalent to the derivation of *false* from *A* and $A \rightarrow \textit{false}$.

Clauses can be represented in different forms

Conditional form. e.g. $D \wedge B \rightarrow E \vee C$.

Disjunctive form. e.g. $\neg D \vee \neg B \vee E \vee C$ where $\neg D$ and $\neg B$ are *negative literals*.
(A *literal* is an atom or the negation of an atom.)

Set of literals form. e.g. $\{\neg D, \neg B, E, C\}$.

Resolution was originally defined by J. Alan Robinson (1965) for clauses that are represented as sets of *literals*, in which the following logical equivalences are “built-in”:

$$\begin{array}{lll} A \vee A & \text{is equivalent to} & A \\ A \vee B & \text{is equivalent to} & B \vee A \\ A \vee (B \vee C) & \text{is equivalent to} & (A \vee B) \vee C. \end{array}$$

The *resolvent* of two clauses represented as sets:

$$\{A\} \cup F \text{ and } \{\neg A\} \cup G$$

is the clause: $F \cup G$.

Unification and factoring

In the non-propositional case, resolution needs to be extended with unification, to make the two atoms resolved upon identical. Given:

$$\begin{array}{c} D \rightarrow E \vee A_1 \\ \text{-----} \\ A_2 \wedge B \rightarrow C \end{array}$$

such that A_1 and A_2 are unifiable, the *resolvent* is:

$$D' \wedge B' \rightarrow E' \vee C'$$

where B' , C' , D' and E' are obtained by applying the most general unifier of A_1 and A_2 to B , C , D and E respectively.

Unification and factoring

More generally and for the set representation of clauses:

Given clauses $X \cup Y$ and $V \cup W$ and a most general unifier θ such that $X\theta = \{A\}$ and $V\theta = \{\neg A\}$ for some atom A , the *resolvent* is $Y\theta \cup W\theta$.

Note a substitution θ is a mapping of variables into terms. A most general unifier is the most general substitution that unifies two expressions, making them identical.

For example, the substitution $\{X \rightarrow a, Y \rightarrow b\}$ unifies $p(X, b)$ and $p(a, Y)$.
 $\{X \rightarrow a, Y \rightarrow b, Z \rightarrow b\}$ unifies $p(X, Z)$ and $p(a, Y)$,
but $\{X \rightarrow a, Y \rightarrow Z\}$ is the most general unifier of $p(X, Z)$ and $p(a, Y)$.

$p(X, g(X))$ and $p(f(Y), Y)$ have no unifier.

We write $E\theta$ for the result of applying the substitution θ to E .

Factoring

Resolution: Given clauses $X \cup Y$ and $V \cup W$ and a most general unifier θ such that $X \theta = \{A\}$ and $V \theta = \{\neg A\}$ for some atom A , the resolvent is $Y \theta \cup W \theta$.

X and V can contain several literals.
 θ factors these literals into one literal.

Example: john shaves everyone who does not shave himself,
john shaves no one who does shave himself. i.e.

$$\begin{aligned} \neg shaves(X, X) &\rightarrow shaves(john, X) \\ shaves(X, X) &\rightarrow \neg shaves(john, X) \end{aligned}$$

In clausal form (renaming variables, to avoid confusion)

$$\begin{aligned} \{shaves(X, X), shaves(john, X)\} \\ \{\neg shaves(Y, Y), \neg shaves(john, Y)\} \end{aligned}$$

The resolvent obtained using the most general unifier $\{X \rightarrow john, Y \rightarrow john\}$ is the empty clause, showing that the clauses are inconsistent.

Reasoning in classical logic involves two kinds of inference rules:

- 1) Inference rules included in resolution +
- 2) Inference rules needed to convert sentences into clausal form.

- 1) Unification in resolution is a more efficient version of the inference rule:

Derive $P(t)$ from $\forall X P(X)$ for any term t .

Resolution also includes backward reasoning, forward reasoning and many other inferences.

- 2) Converting sentences into clausal form involves the use of many other inference rules, including:

$A \rightarrow B$ is equivalent to $\neg A \vee B$
 $\neg(A \vee B)$ is equivalent to $\neg A \wedge \neg B$

For example, $A \rightarrow (B \rightarrow C)$ can be converted into $A \wedge B \rightarrow C$

$(A \rightarrow B) \rightarrow C$ can be converted into $A \vee C$ and $B \rightarrow C$

Conversion into clausal form eliminates existential quantifiers

$\exists Y \forall X \text{ likes}(Y, X)$ is converted into $\text{likes}(f, X)$
where f is a constant that occurs nowhere else.

$\forall X \exists Y \text{ likes}(Y, X)$ is converted into $\text{likes}(f(X), X)$
where f is a function symbol that occurs nowhere else.

Note that in English “*someone likes every one*” is ambiguous.
Is there a natural way to express the two different meanings in Japanese, Chinese or Vietnamese?

$\forall X (X \subseteq Y \leftarrow \forall Z (Z \in X \rightarrow Z \in Y))$ is converted into

$X \subseteq Y \vee \text{arbitrary}(X, Y) \in X$
 $X \subseteq Y \leftarrow \text{arbitrary}(X, Y) \in Y$

Note that resolution treats these two clauses as a procedure:
To show $X \subseteq Y$, assert $\text{arbitrary}(X, Y) \in X$ and show $\text{arbitrary}(X, Y) \in Y$.

Resolution treats $X \subseteq Y \vee \text{arbitrary}(X,Y) \in X$ and $X \subseteq Y \leftarrow \text{arbitrary}(X,Y) \in Y$ as a procedure:

Given $\text{friends} = \{\text{mary}, \text{john}\}$ $\text{club} = \{\text{mary}, \text{john}, \text{bob}\}$

To show $\text{friends} \subseteq \text{club}$ derive *false* from

$\neg \text{friends} \subseteq \text{club}$ (0)

$X \subseteq Y \vee \text{arbitrary}(X,Y) \in X$ (1)

$X \subseteq Y \vee \neg \text{arbitrary}(X,Y) \in Y$ (2)

$X \in \text{friends} \rightarrow X = \text{mary} \vee X = \text{john}$ (3)

$X \in \text{club} \leftarrow X = \text{mary}$ (4a)

$X \in \text{club} \leftarrow X = \text{john}$ (4b)

$X \in \text{club} \leftarrow X = \text{bob}$ (4c)

$\text{arb}(\text{friends}, \text{club}) \in \text{friends}$ (5) from (0) & (1)

$\neg \text{arb}(\text{friends}, \text{club}) \in \text{club}$ (6) from (0) & (2)

$\text{arb}(\text{friends}, \text{club}) = \text{mary} \vee \text{arb}(\text{friends}, \text{club}) = \text{john}$ (7) from (3) & (5)

$\neg \text{arb}(\text{friends}, \text{club}) = \text{mary}$ (8a) from (4a) & (6)

$\neg \text{arb}(\text{friends}, \text{club}) = \text{john}$ (8b) from (4b) & (6)

false from (7) & (8a) & (8b)

Comparison with logic programs

The definition of subset can be represented as a logic program.

$\forall X (X \subseteq Y \leftarrow \forall Z (Z \in X \rightarrow Z \in Y))$ can be translated into the program:

$$X \subseteq Y \leftarrow \neg \exists Z (Z \in X \wedge \neg Z \in Y)$$

Show every set is a subset of the *universe*, where the *universe* is defined by the clause: $X \in \textit{universe}$. Let *arb* be a constant symbol about which nothing is known, so it represents any set.

Goal: $\textit{arb} \subseteq \textit{universe}$

Subgoal: $\neg \exists Z (Z \in \textit{arb} \wedge \neg Z \in \textit{universe})$

Naf: $Z \in \textit{arb} \wedge \neg Z \in \textit{universe}$

Note any subgoal can be selected.

Naf: $Z \in \textit{universe}$

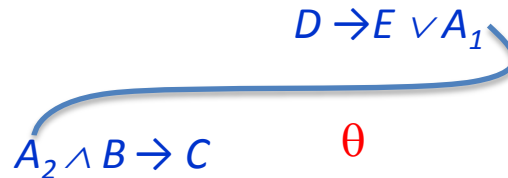
Succeed

Failure because one of the subgoals fails.

Succeed

The Connection Graph Proof Procedure

The [connection graph proof procedure](#) (Kowalski 1974) implements resolution, by pre-computing links between the conditions and conclusions of clauses.



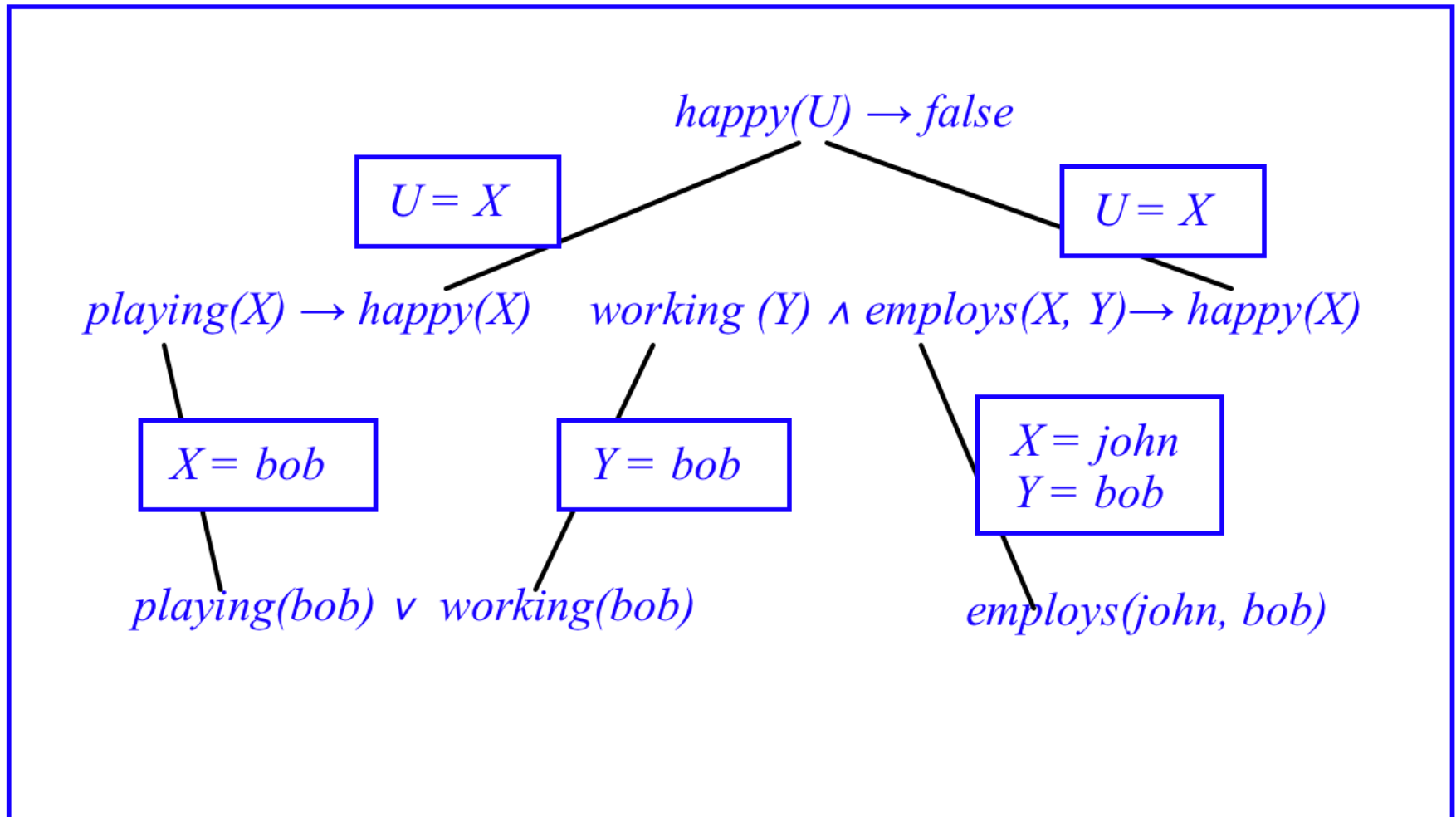
It labels links with their unifying substitutions θ .
These links can then be activated later when the need arises.

Any strategy can be used for selecting and activating links, including forwards and backwards reasoning, generating resolvent clauses, whose new links are inherited from their parent clauses.

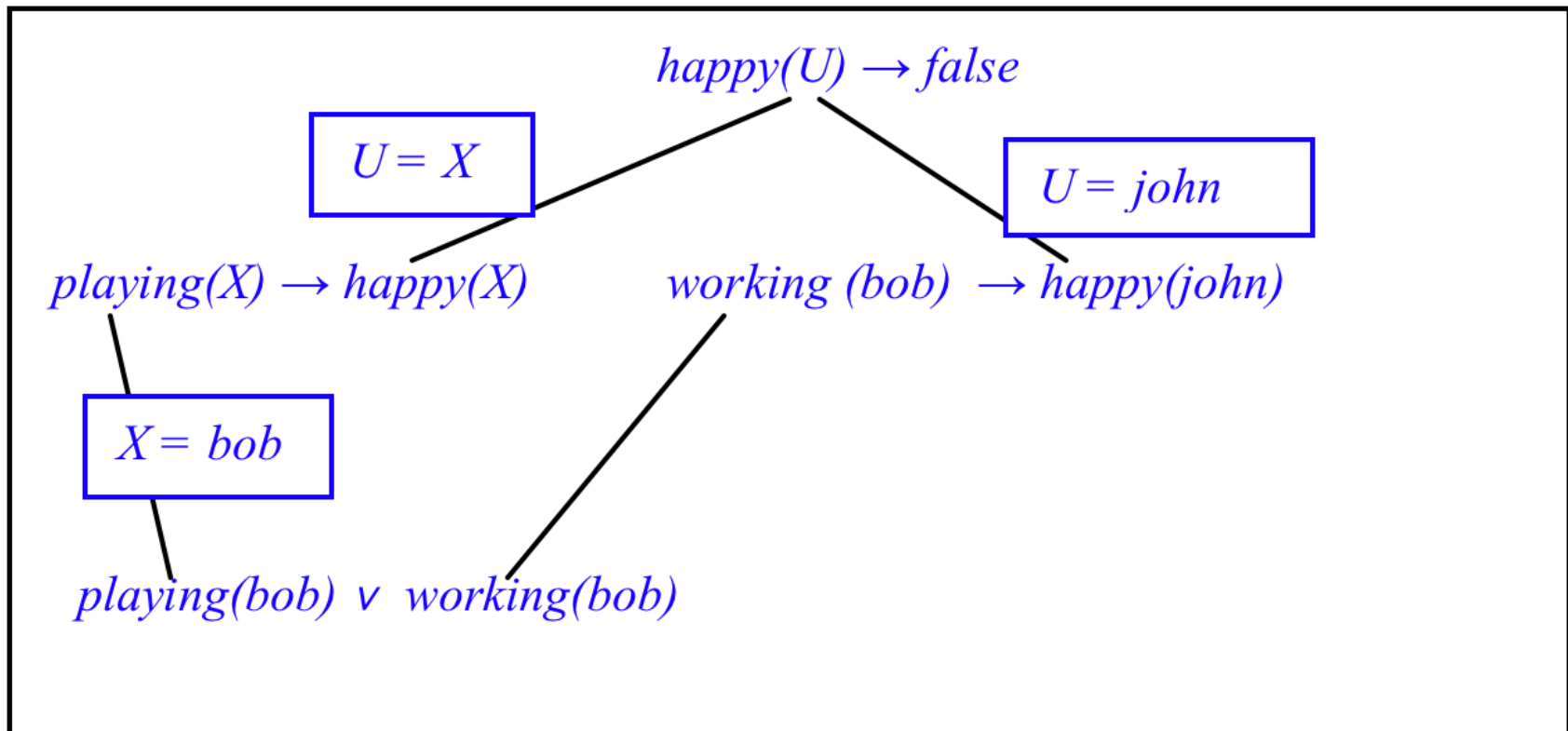
In many cases, parent clauses can be deleted or over-written, when all their links have been activated.

Links that are activated frequently can be compiled into shortcuts, which achieve the same effects more directly, like heuristic rules and stimulus-response associations.

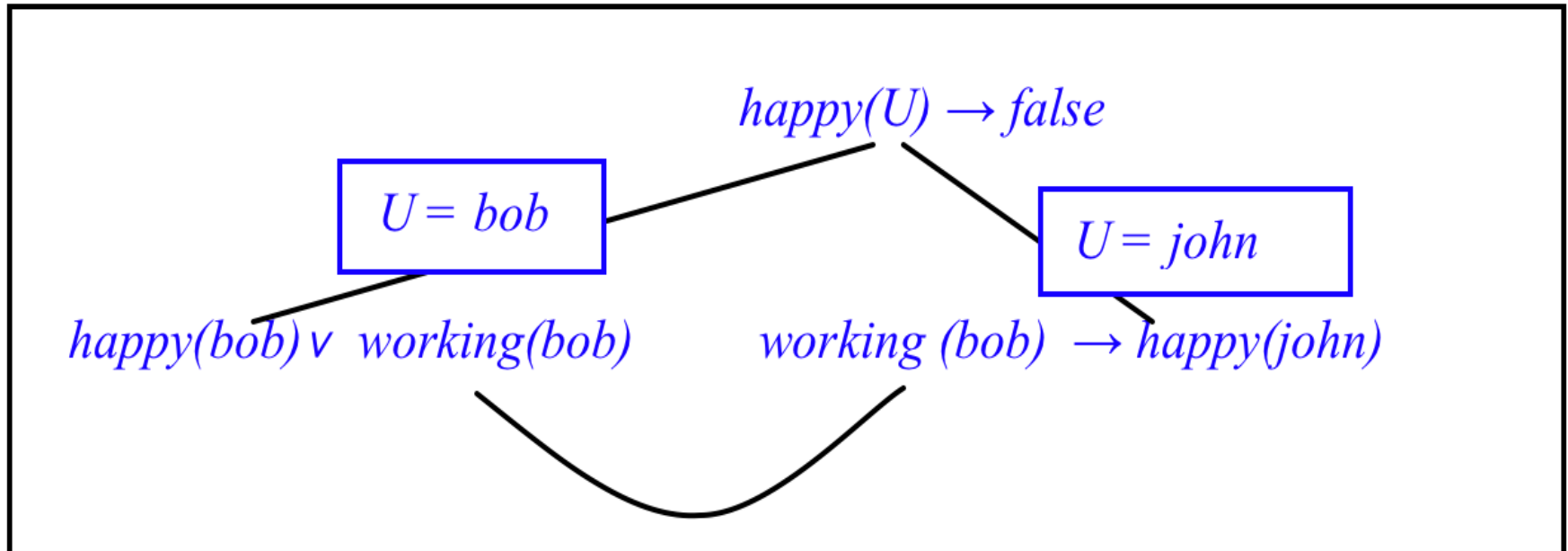
Connection graphs implement resolution with most general unifiers pre-computed



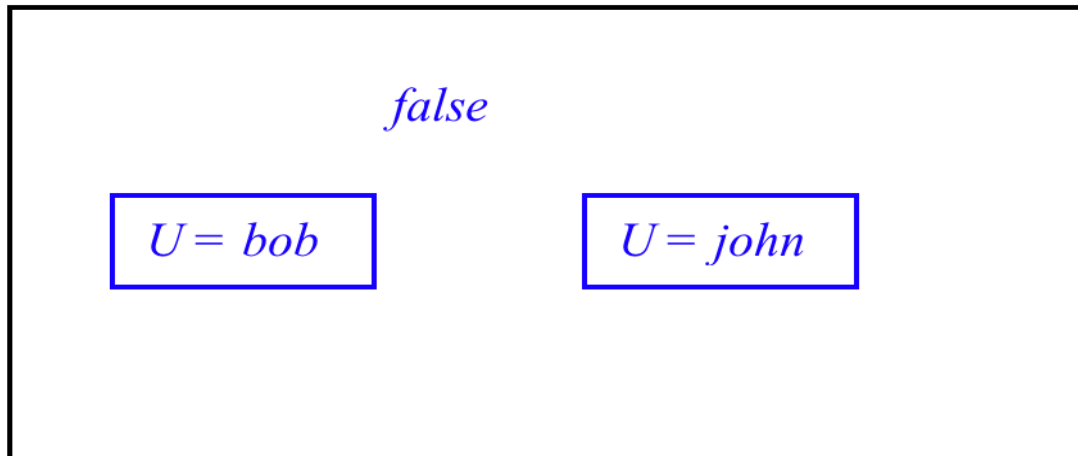
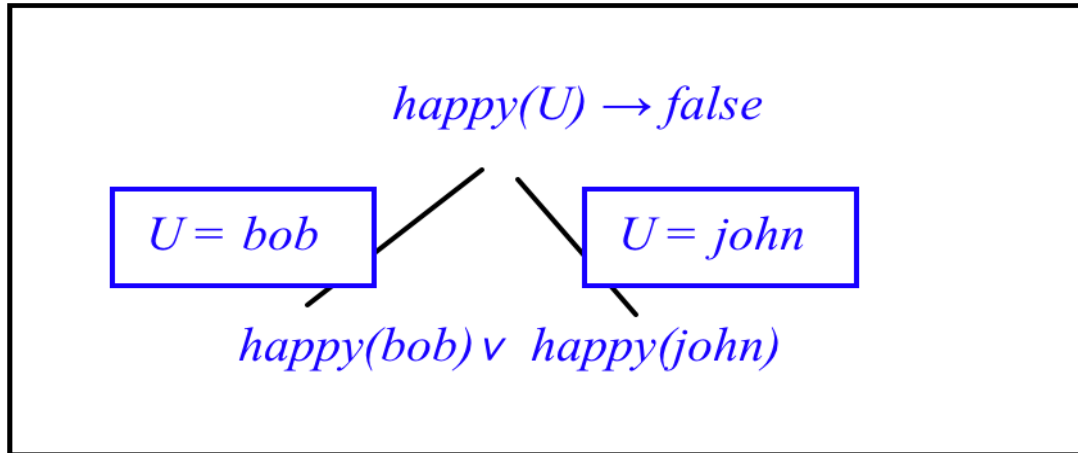
Clauses with unlinked atoms can be deleted, so that resolvents sometimes replace their parents.
New links can be computed from parent links.



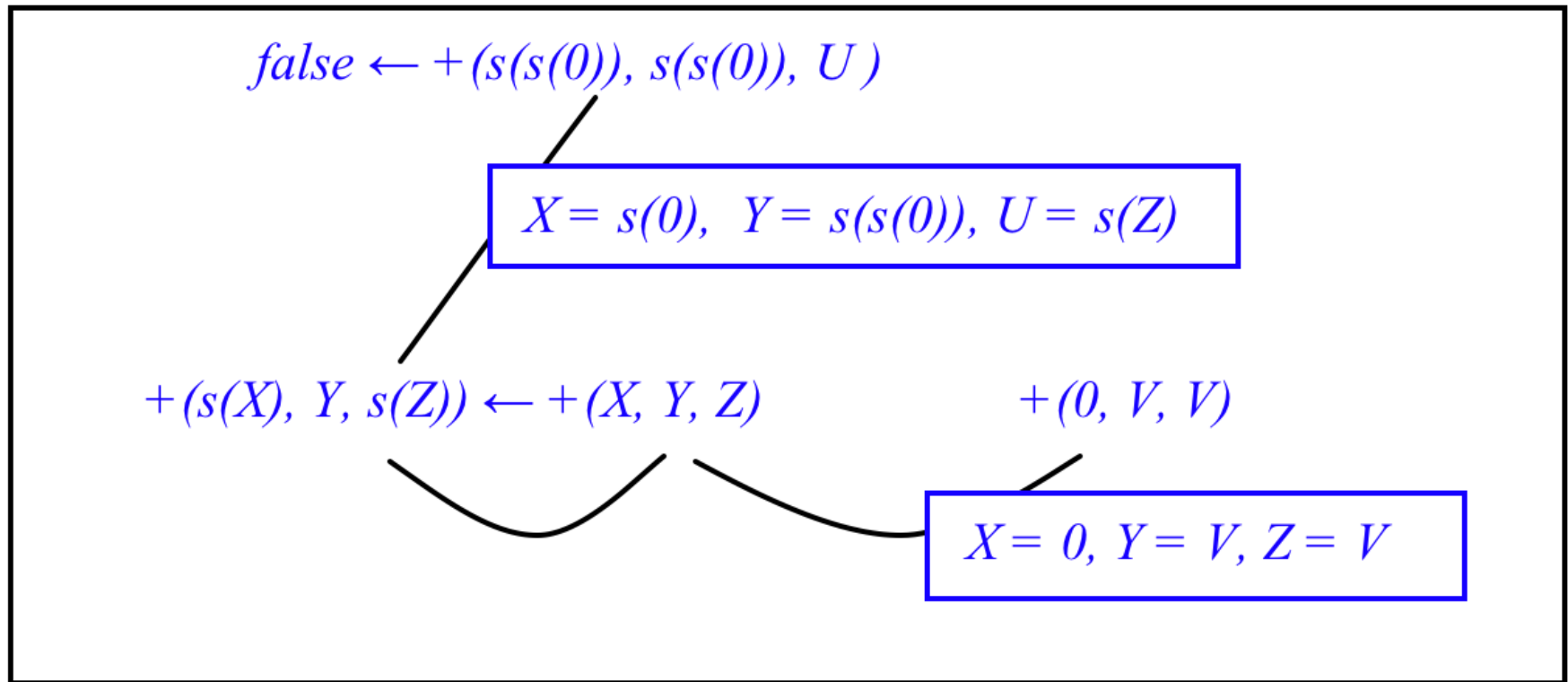
In this example, the activation of links generalises forward reasoning



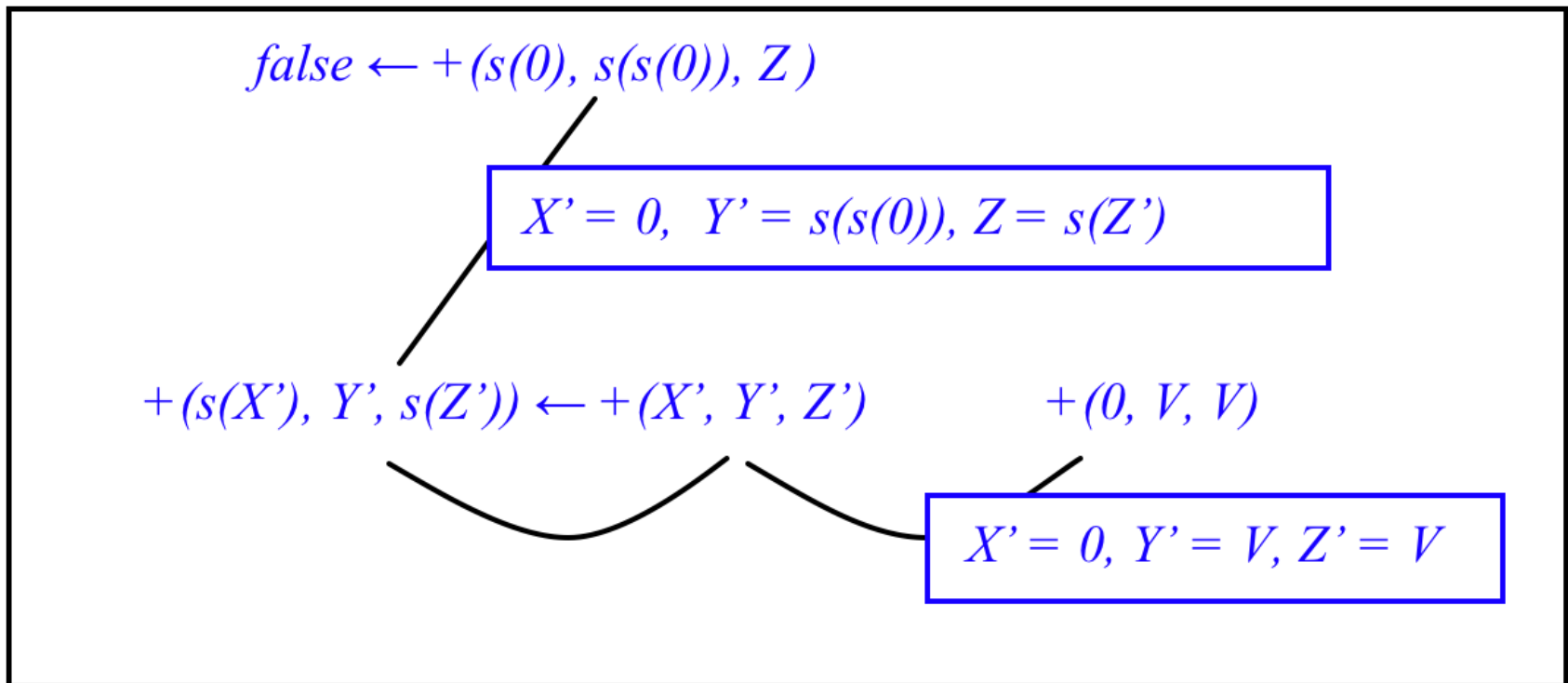
Links can be activated in parallel



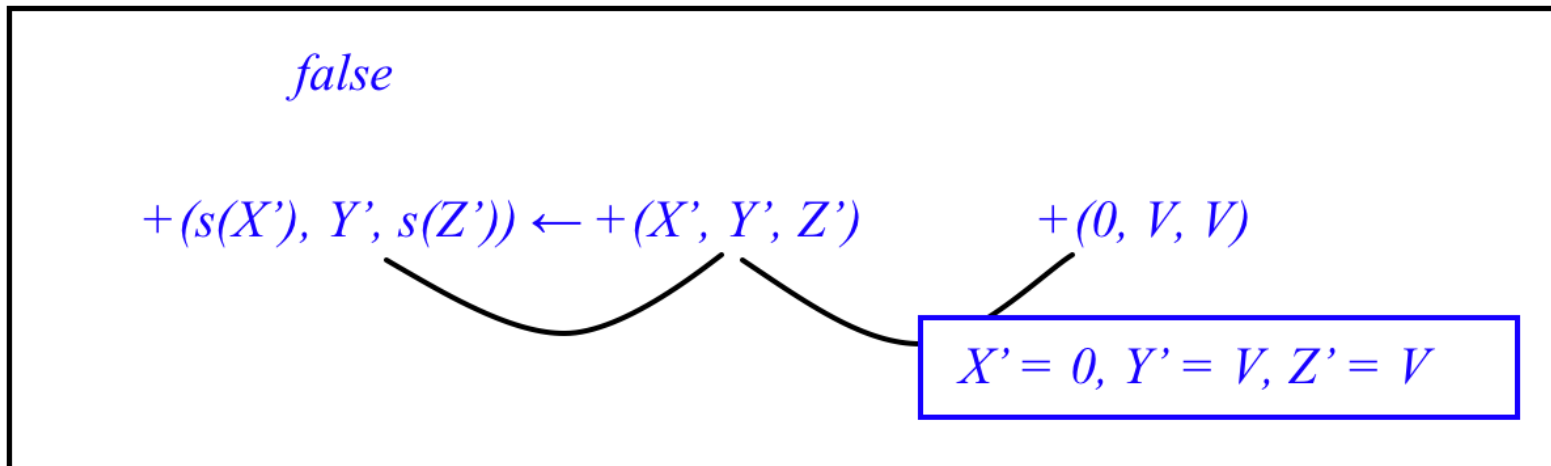
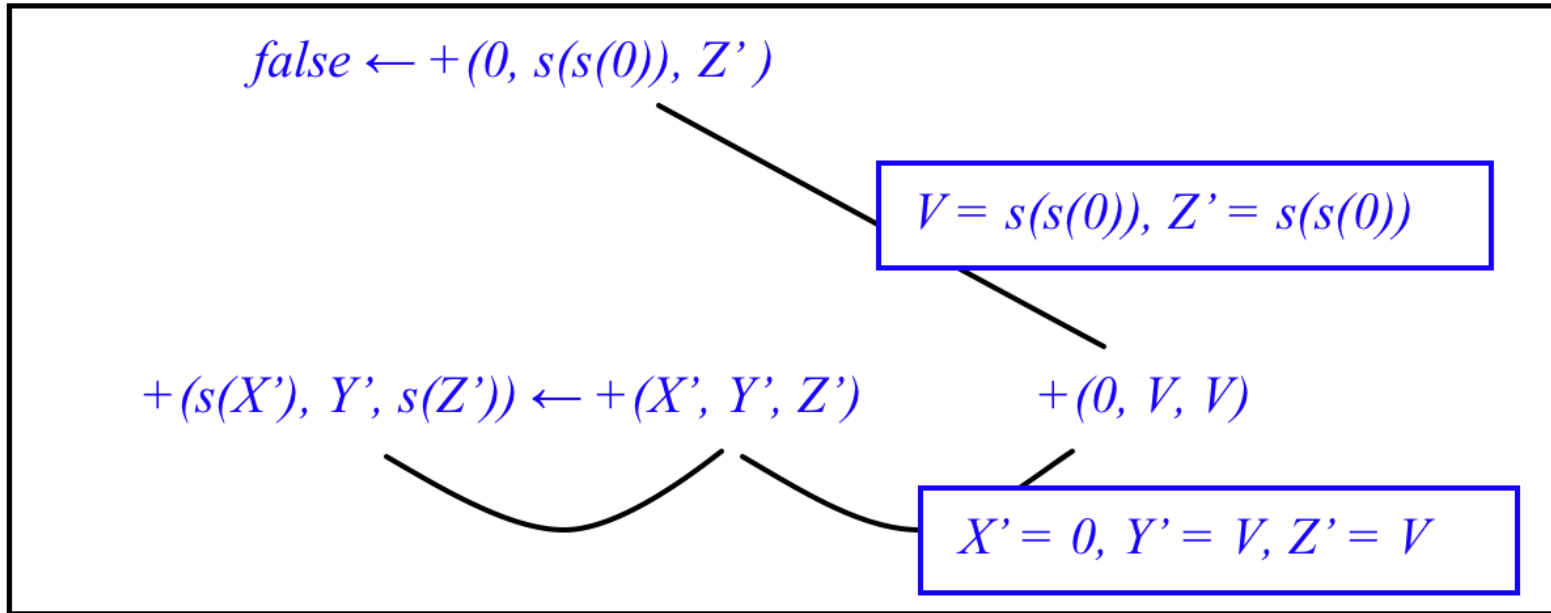
Self-resolving clauses contain internal links



Backward reasoning generates a new goal clause and deletes the parent goal clause, as in tail-recursion optimisation



The cumulative substitutions $U = s(Z)$, $Z = s(Z')$, $Z' = s(s(0))$ compute the sum $U = s(s(s(s(0))))$.



Connection graphs can combine logic, search, connectionism, learning and decision making

- Links can be weighted by statistics about how often they have contributed to successful outcomes in the past.
- Different input observations and goals can be assigned different strengths (or utilities).
- The strength of observations and goals can be propagated throughout the graph in proportion to the weights on the links. Activating links with the current highest weighted strength implements a form of best-first search, and is similar to the activation networks of [Maes, 1990].

Homework 10

1. Use the logical equivalences:

$$A \rightarrow B \quad \text{is equivalent to} \quad \neg A \vee B \quad (1)$$

$$\neg(A \vee B) \quad \text{is equivalent to} \quad \neg A \wedge \neg B \quad (2)$$

$$(A \wedge B) \vee C \quad \text{is equivalent to} \quad (A \vee C) \wedge (B \vee C) \quad (3)$$

$$\neg\neg A \quad \text{is equivalent to} \quad A \quad (4)$$

$$(A \vee (B \vee C)) \quad \text{is equivalent to} \quad (A \vee B) \vee C \quad (5)$$

$$\neg(A \wedge B) \quad \text{is equivalent to} \quad \neg A \vee \neg B \quad (6)$$

to show that

(a) $A \rightarrow (B \rightarrow C)$ is equivalent to $A \wedge B \rightarrow C$

(b) $(A \rightarrow B) \rightarrow C$ is equivalent to $A \vee C$ and $B \rightarrow C$

2. Use the clausal form of the definition of subset, to show that

(a) the empty set is a subset of every set,
where the empty set is defined by $X \in \text{empty} \rightarrow \text{false}$

(b) every set is a subset of the universe.
where the universe is defined by $X \in \text{universe}$

Homework 10

3. Use backward reasoning and negation as failure with the logic program:

$$X \subseteq Y \leftarrow \neg \exists Z (Z \in X \wedge \neg Z \in Y)$$

to show that:

(a) $friends \subseteq club$ where

$mary \in friends$

$john \in friends$

$mary \in club$

$john \in club$

$bob \in club$

Note that to show a goal fails, it is necessary to show that all ways of trying to solve the goal fail.

(b) the empty set is a subset of every set, where the empty set is represented by a constant $empty$, and there are no clauses of the form $t \in empty$, for any term t .

Lecture 11 Chapter 10 Abduction

Abduction is the task of generating assumptions Δ to explain observations O .

For example, if instead of observing fire,
I observe *there is smoke*, and
I believe *there is smoke if there is a fire*.

Backwards reasoning from the observation
generates an assumption *there is a fire*.

Abduction in the agent cycle

Maintenance goal: *if there is an emergency then I get help.*

Beliefs: *a person gets help if the person alerts the driver.*

*a person alerts the driver
if the person presses the alarm signal button.*

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

Observation:

Abduction by backward reasoning:

Forward reasoning, new belief:

Forward reasoning, achievement goal:

Backward reasoning, subgoal:

Backward reasoning, action:

there is smoke

there is fire.

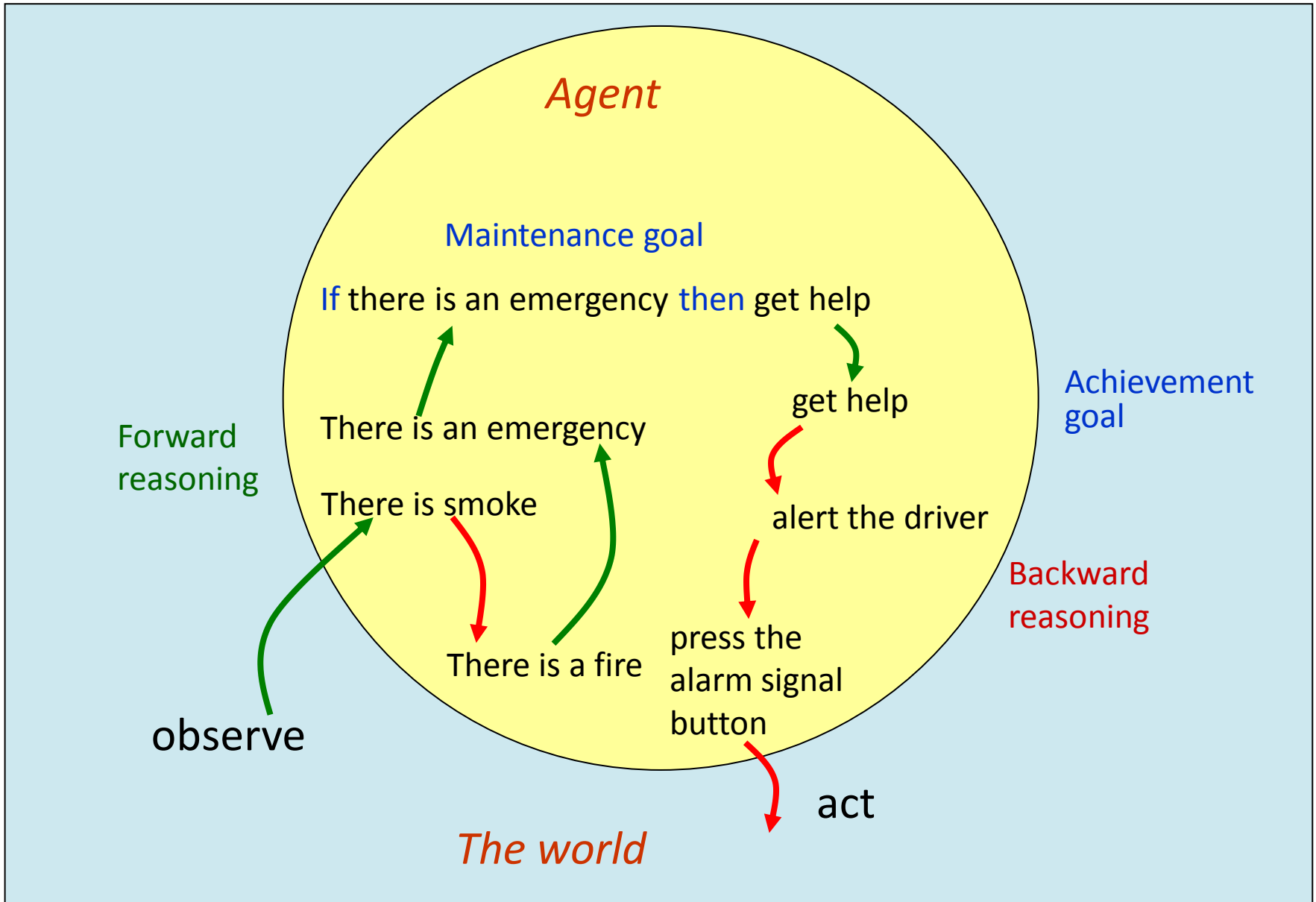
there is an emergency.

I get help!

I alert the driver!

I press the alarm signal button!

Abduction on the London underground



The term *abduction* was introduced by the logician Charles Sanders Peirce (1931)

He gave the following example:

Deduction: *All the beans from this bag are white.*
 These beans are from this bag.
 Therefore *These beans are white.*

Induction: *These beans are from this bag.*
 These beans are white.
 Therefore *All the beans from this bag are white.*

Abduction: *All the beans from this bag are white.*
 These beans are white.
 Therefore *These beans are from this bag.*

There can be several alternative abductive explanations of the same observation

Beliefs:

the grass is wet if it rained.

the grass is wet if the sprinkler was on.

the grass is wet is a closed predicate, which is completely defined (so the closed world assumption can be applied)

it rained and *the sprinkler was on* are open predicates, which are undefined.

Observation:

the grass is wet.

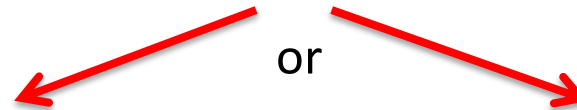
Backward reasoning:

or

Hypotheses:

it rained.

the sprinkler was on.



There can be several alternative abductive explanations of the same observation. The more observations explained the better.

Beliefs:

the grass is wet if it rained.

the grass is wet if the sprinkler was on.

Observation:

the grass is wet.

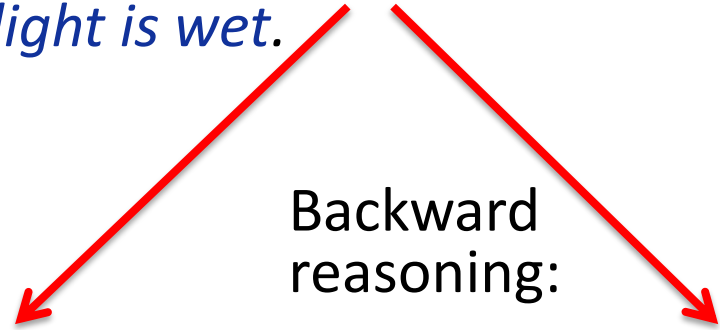
Additional observation:

the skylight is wet.

Forward
reasoning:



Backward
reasoning:



Hypotheses:

it rained.

the sprinkler was on.

Different hypotheses can also have different probabilities.

In Dubai, it would be more probable that the sprinkler was on.

In Kyoto, it would be more probable that it rained.

Abductive hypotheses should be relevant to the observation

Relevance is automatically satisfied by reasoning backwards .

Backward reasoning ensures every hypothesis is relevant to the observation.

Relevance is weaker than the requirement that explanations be minimal.

Minimality insists that no subset of the explanation is also an explanation.

Beliefs: *the floor is wet if it rained and the window was open.*
 the floor is wet if it rained and there is a hole in the roof.
 there is a hole in the roof.

Observation: *the floor is wet.*
Relevant explanation: *it rained and the window was open.*
Minimal explanation: *it rained.*
Irrelevant explanation: *it rained and the dog was barking.*

Abductive hypotheses should be consistent with existing beliefs.

The consistency requirement excludes impossible explanations, such as *it rained*, if there were clothes outside and they didn't get wet.

not wet can be represented by the positive concept *dry*.

Constraint:
i.e. *if a thing is dry and the thing is wet then false.*
nothing is both dry and wet.

Belief: *the clothes outside are wet if it rained.*
Additional observation: *the clothes outside are dry.*

Hypothesis: *it rained*
Forward reasoning: *the clothes outside are wet*
Forward reasoning: *if the clothes outside are dry then false*
Forward reasoning: *false*

The derivation of *false* eliminates the hypothesis *it rained* as a candidate explanation of the observation *the grass is wet*.

Contraries and strong negation

Many concepts occur as pairs of *contrary* positive concepts, like *wet* and *dry*, *tall* and *short*, *big* and *small*, *good* and *bad*.

Often these contraries are expressed as negations, as in *not wet* instead of *dry* and *not dry* instead of *wet*. This use of negation is sometimes called *strong negation*.

Viewed as a form of negation, it has the *truth value gap property* that there can be instances of a predicate that are neither *true* nor *false*. For example, if my clothes are merely damp, I might consider them as being neither wet nor dry.

Contrary predicates with truth gaps are a natural way to represent vague concepts.

For every pair of contrary predicates, we have constraints of the form:

if predicate and contrary-predicate then false.

Global warming

Beliefs: *world temperatures rise
if there is a man-made increase in greenhouse gases.*

*world temperatures rise
if there is a natural cyclic increase.*

Observation: *world temperatures are rising*

Backward
reasoning:

Hypotheses: *there is a man-made
increase in greenhouse gases.*

*there is a natural
cyclic increase*

We may prefer to believe a hypothesis that is judged to have greater probability.

According to expert opinion, most of the observed increase in global temperatures since the mid-20th century is more than 90% likely to be due to the increase in man-made greenhouse gas concentrations.

But the ultimate test is what are the consequences of the hypotheses, and how do the consequences affect our maintenance goals.

Abduction can be used for fault diagnosis

Beliefs: *car doesn't start*
if fault in the battery.

car doesn't start
if fault in the fuel supply.

Observation:

car doesn't start

Backward
reasoning:

Hypotheses:

fault in the battery

fault in the fuel supply

We may prefer to believe a hypothesis that is judged to have greater probability, judged by using statistics about the past.

But the ultimate test is what are the consequences of the hypotheses, and how do the consequences affect our maintenance goals. For example, how easy is it to check the hypotheses by additional observations.

In abductive logic programming (ALP) agents observations O and goals G are treated the same.

Given beliefs B , goals G and observations O ,
The purpose of an agent's life is to generate a set Δ
of actions and assumptions about the world such that:

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

G : *if there is an emergency then I get help.*

B : *a person gets help if the person alerts the driver.
a person alerts the driver
if the person presses the alarm signal button.
there is an emergency if there is a fire.
there is smoke if there is a fire.*

O : *there is smoke*

$G \cup O$ is *true* in the minimal model of $B \cup \Delta$
where $\Delta = \{$ *there is a fire, I press the alarm button* $\}$

ALP agents combine abduction and decision making

There can be several, alternative Δ that, together with B , make G and O both *true*.

The challenge is to find the best Δ within the computational resources available.

In **classical decision theory**, the value of an action is measured by the expected utility of its consequences.

In **philosophy of science**, the value of an explanation is measured similarly in terms of its probability and explanatory power. (The more observations explained the better.)

In ALP agents, the same measure can be used to evaluate both candidate actions and candidate explanations.

Homework 11

Given

G : *if there is an emergency then I get help.*

B : *a person gets help if the person alerts the driver.
a person alerts the driver
if the person presses the alarm signal button.
there is an emergency if there is a fire.
there is smoke if there is a fire.*

O : *there is smoke*

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

1. What is the minimal model of $B \cup \Delta$?
2. Show that $G \cup O$ is *true* in the minimal model of $B \cup \Delta$.

Lecture 12 Chapter 11 The Prisoner's Dilemma

The prisoner's dilemma

The dilemma about whether or not to take an umbrella

Classical decision theory

The role of decision making in Computational Logic

Smart Choices involve paying greater attention to goals

The Prisoner's Dilemma in Classical Decision Theory:

There are two prisoners, bob and john –

Both are arrested and questioned

Candidate actions: *I cooperate or I refuse to cooperate*

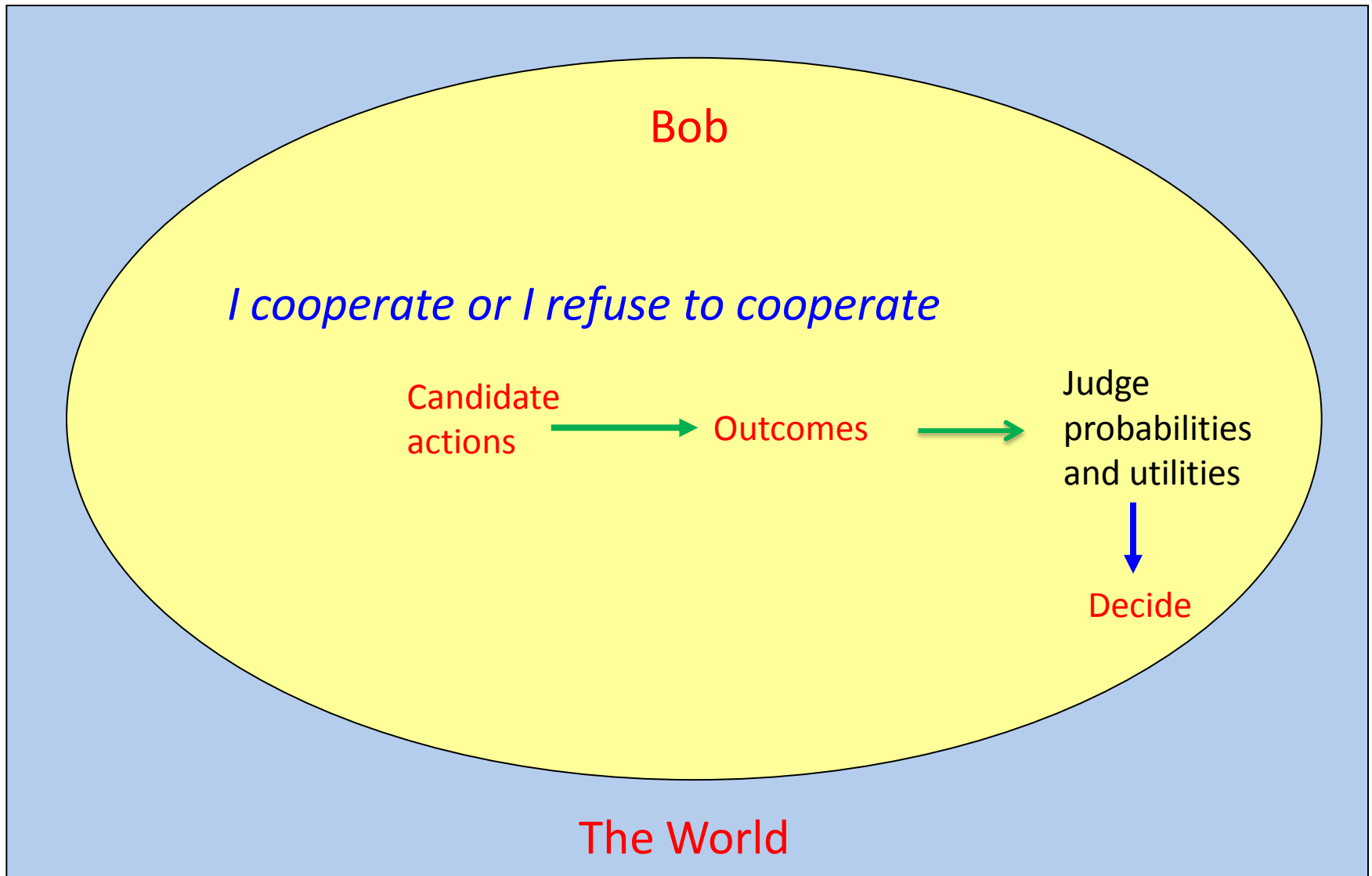
Beliefs: *A prisoner gets 0 years in jail
if the prisoner cooperates
and the other prisoner refuses to cooperate*

*A prisoner gets 4 years in jail
if the prisoner refuses to cooperate
and the other prisoner cooperates*

*A prisoner gets 3 years in jail
if the prisoner cooperates
and the other prisoner cooperates*

*A prisoner gets 1 year in jail
if the prisoner refuses to cooperate
and the other prisoner refuses to cooperate*

Prisoner's dilemma viewed in Decision-theoretic terms



A simpler, but similar example: The dilemma of whether to take an umbrella

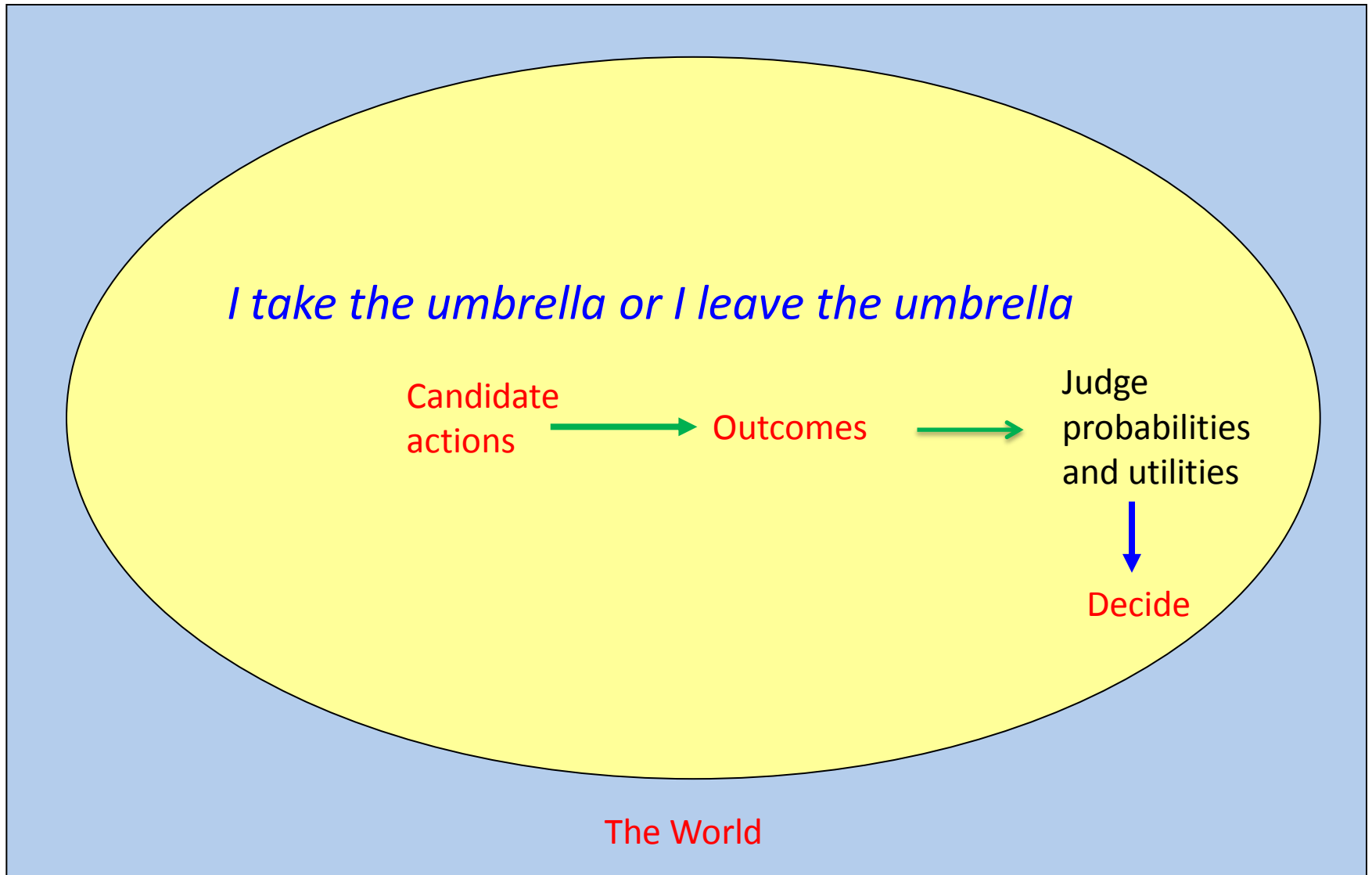
Candidate actions: *I take the umbrella or I leave the umbrella.*

Beliefs: *I stay dry if I take the umbrella.
I get wet if I leave the umbrella and it rains.
I stay dry if it does not rain.*

Assume (as in abduction) *I take the umbrella .*
Outcomes : *I stay dry if it rains.
I stay dry if it does not rain.*

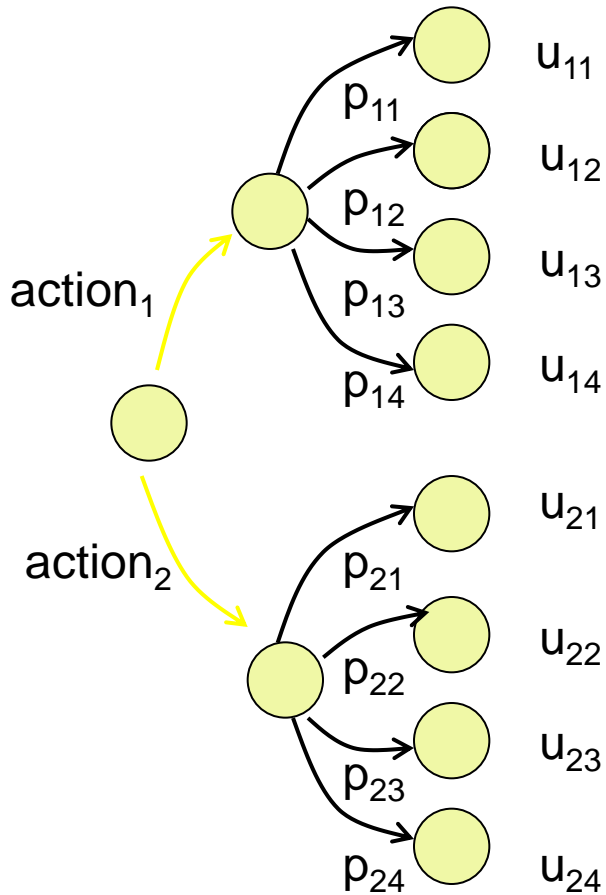
Assume (as in abduction) *I leave the umbrella .*
Outcomes : *I get wet if it rains.
I stay dry if it does not rain.*

Umbrella dilemma viewed in Decision-theoretic terms



Decision Theory:

to find the expected utility of a proposed action,
find all the alternative resulting states of affairs,
weigh the utility of each such state by its probability, and
add them all up.



Expected utility of action₁

$$p_{11} \cdot u_{11} + p_{12} \cdot u_{12} + p_{13} \cdot u_{13} + p_{14} \cdot u_{14}$$

Expected utility of action₂

$$p_{21} \cdot u_{21} + p_{22} \cdot u_{22} + p_{23} \cdot u_{23} + p_{24} \cdot u_{24}$$

Choose the action of highest expected utility

Deciding whether or not to carry an umbrella

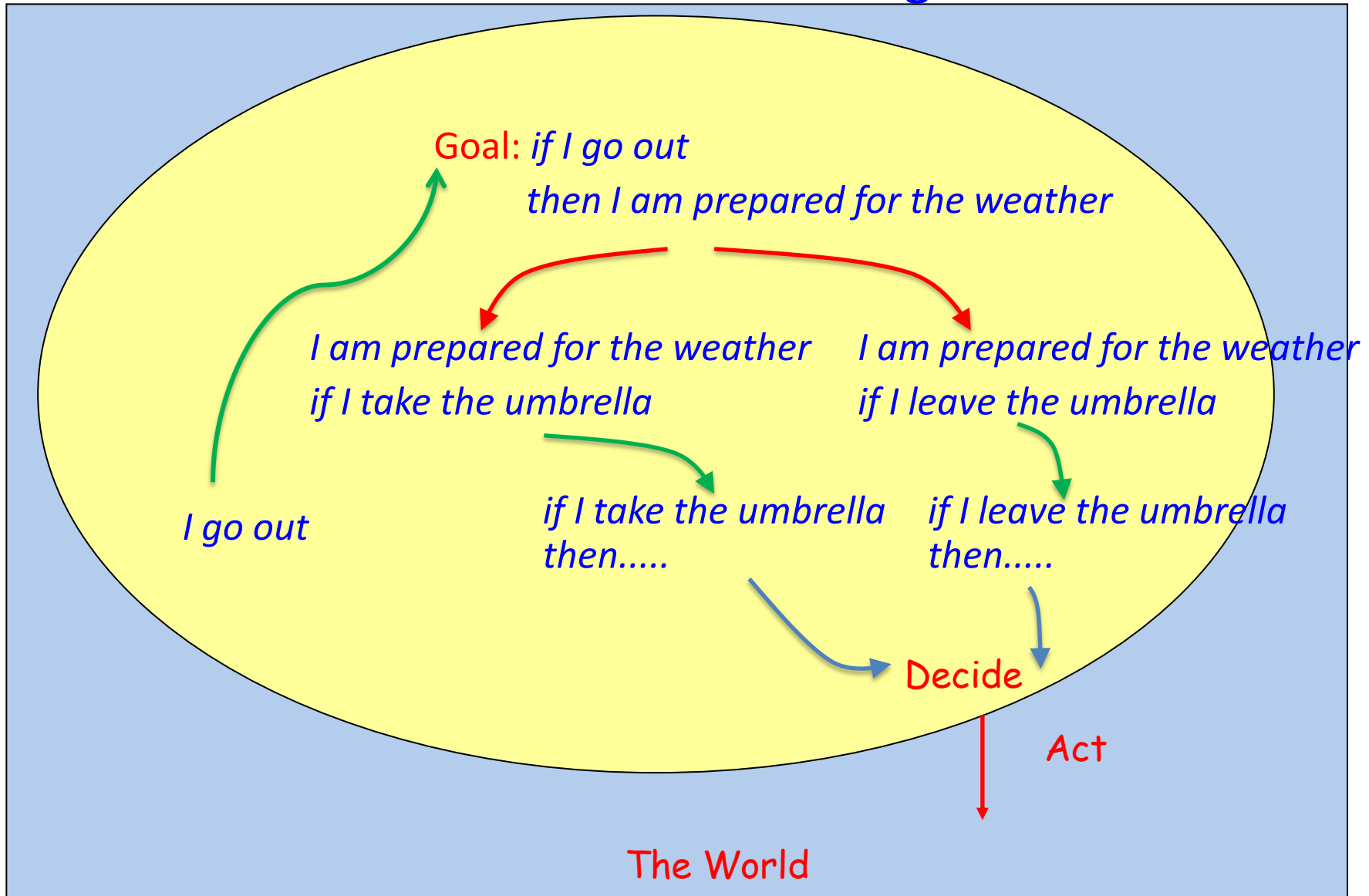
Assume	Probability it rains	= .1
	Probability it does not rain	= .9
	Utility of getting wet	= -10
	Utility of staying dry	= 1
	Utility of taking the umbrella	= -2
	Utility of leaving the umbrella	= 0

Assume I take the umbrella .
Forward reasoning I stay dry with probability 1 and utility -2 +1.
Expected utility = -1

Assume I leave the umbrella .
Forward reasoning I get wet with probability .1 and utility -10.
I stay dry with probability .9 and utility 1
Expected utility $0 - 10 \cdot .1 + 1 \cdot .9 = -1 + .9 = -.1$

Decide I do not carry an umbrella!

The umbrella dilemma viewed in logical terms



A more practical alternative is to use lower level maintenance goals (or heuristic rules) instead:

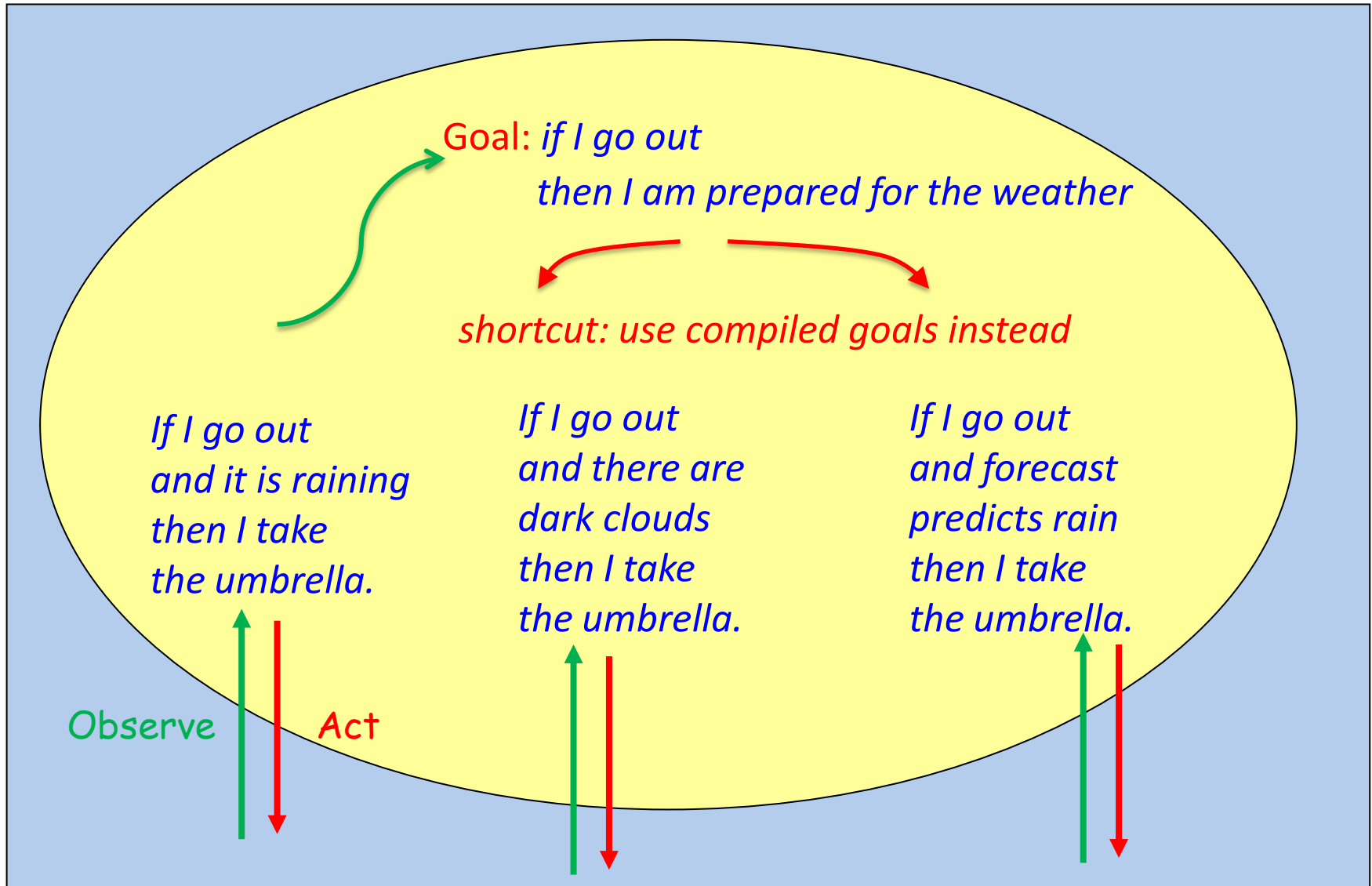
*If I go out and it is raining
then I take an umbrella.*

*If I go out and there are dark clouds in the sky
then I take an umbrella.*

*If I go out and the weather forecast predicts rain
then I take an umbrella.*

The lower level goals compile decision-making into the thinking component of the agent cycle. The compilation might be an exact implementation of the Decision Theoretic specification. Or it might be only an approximation.

The umbrella dilemma compiled into lower level goals



The Prisoner's Dilemma in Classical Decision Theory:

Candidate actions: *I cooperate or I refuse to cooperate*

Beliefs:

- A prisoner gets 0 years in jail
if the prisoner cooperates
and the other prisoner refuses to cooperate*
- A prisoner gets 4 years in jail
if the prisoner refuses to cooperate
and the other prisoner cooperates*
- A prisoner gets 3 years in jail
if the prisoner cooperates
and the other prisoner cooperates*
- A prisoner gets 1 year in jail
if the prisoner refuses to cooperate
and the other prisoner refuses to cooperate*

Decision making in the prisoner's dilemma

Assume (as in abduction) *I cooperate*
Outcomes derived from assumption
using forward reasoning:

*I get 0 years in jail
if john refuses.*

*I get 3 years in jail
if john cooperates.*

Assume (as in abduction) *I refuse to cooperate.*
Outcomes derived from assumption
using forward reasoning:

*I get 4 years in jail
if john cooperates.*

*I get 1 year in jail
if john refuses.*

In Classical Decision Theory

Assume Probability john cooperates with the police = .5
Probability john refuses to cooperate = .5
Utility = N, where N is the number of years I go to jail.

Assume I cooperate

Utility if john refuses and I get 0 years in jail = 0

Utility if john cooperates and I get 3 years in jail = 3

Expected utility $.5 \cdot 0 + .5 \cdot 3 = 1.5$

Assume I refuse

Utility if john cooperates and I get 4 years in jail = 4

Utility if john refuses and I get 1 year in jail = 1

Expected utility $.5 \cdot 4 + .5 \cdot 1 = 2.5$

Decide I cooperate with the police!

In Classical Decision Theory

But assume john does the same calculation.

Then john also refuses to cooperate!

Certain outcome: *I get 3 years in jail
because john and I both cooperate with the police.*

If we both refused to cooperate with the police,
we would have gotten only 1 year in jail.

In game theory, the prisoner's dilemma is iterated.
The first prisoner makes a candidate decision,
simulates the other prisoner,
and makes a revised decision,
etc.

Alternatively, the prisoners could evaluate their actions
using different utilities.

In Classical Decision Theory – with a less selfish utility function

Assume Probability john cooperates = .5

Probability john refuses = .5

Utility = $N + M$ where N is the number of years I go to jail and
 M is the number of years john goes to jail.

Assume I cooperate

Utility if john refuses = $0 + 4$

Utility if john cooperates = $3 + 3$

Expected utility $.5 \cdot 4 + .5 \cdot 6$ = 5 years in jail.

Assume I refuse

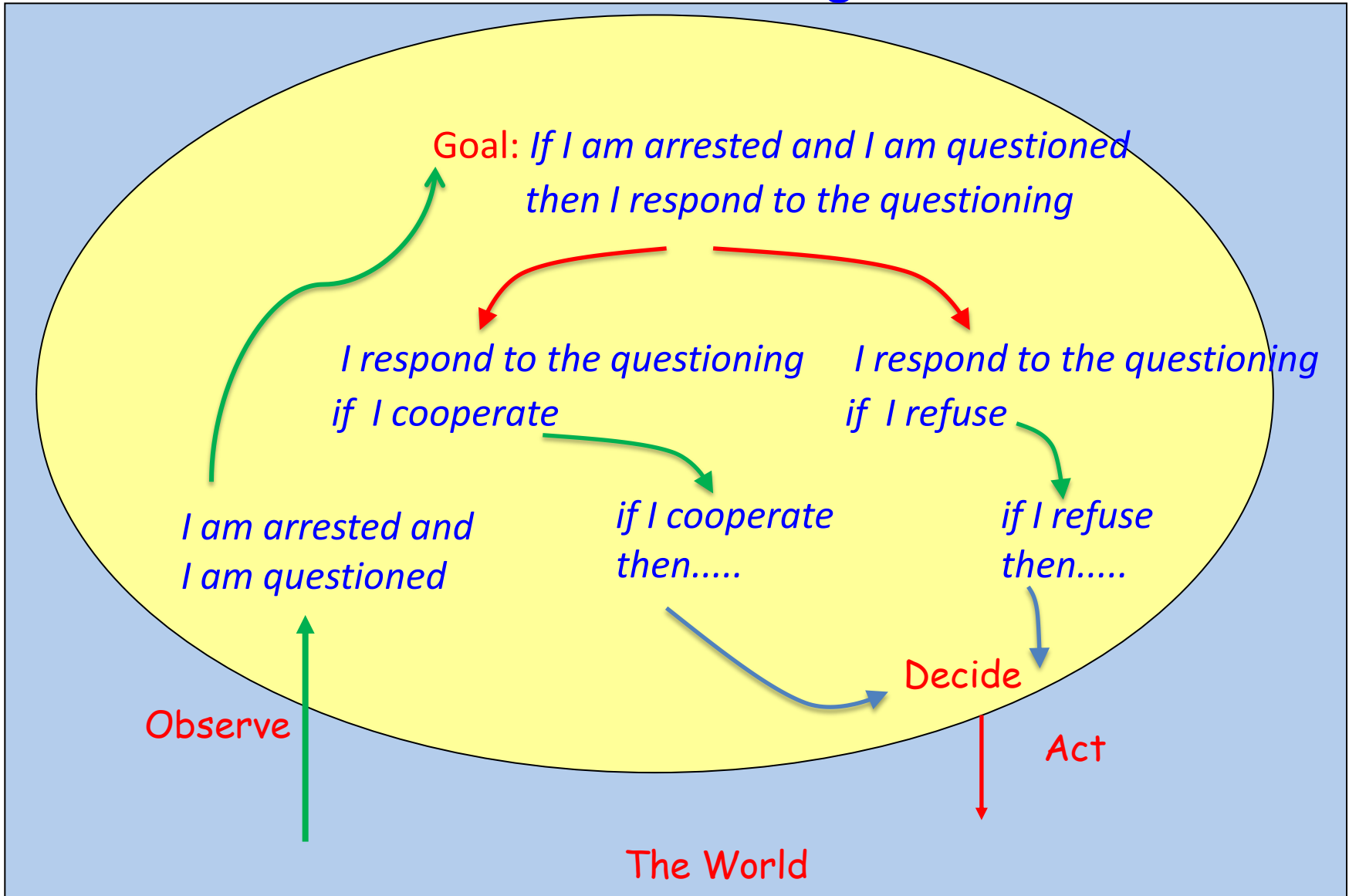
Utility if john cooperates = $4 + 0$

Utility if john refuses = $1 + 1$

Expected utility $.5 \cdot 4 + .5 \cdot 2$ = 3 years in jail

Decide I refuse to cooperate with the police!

Prisoner's dilemma viewed in logical terms



A more practical alternative is to use lower level maintenance goals (or heuristic rules) instead:

*if an agent requests me to perform an action,
and the action does not harm another person
then I perform the action.*

*if an agent requests me to perform an action,
and the action harms another person
then I refuse to perform the action.*

Smart choices – a better decision theory

Classical decision theory assumes that all of the alternative candidate actions to be decided between are given in advance.

But as [Keeney, 1992; Hammond *et al.*, 1999; Carlson *et al.*, 2008]] and other decision analysts point out,

the assumption is not only unrealistic as a description of human decision making, but also unhelpful as a prescription for making better decisions.

To make smart decisions, it is necessary first to identify the goals that motivate the alternatives. These goals might be generated explicitly by higher-level thinking or they might be hidden implicitly in lower-level heuristic rules.

Lecture 13 Chapter A4

Minimal Models and Negation

Review:

Truth, logical consequence, Herbrand model, definite clauses

The semantics of negation as failure

The distinction between operational rules and emergent properties

Stable model semantics for logic programs with negative conditions

Truth versus proof in arithmetic

Truth, logical consequence and definite clauses - review

A *Herbrand interpretation* is a set of ground atomic sentences, representing the set of all ground atomic sentences that are true.

Logic programs have the form: $C_1 \wedge \dots \wedge C_n \wedge \text{not } D_1 \wedge \dots \wedge \text{not } D_m \rightarrow E$
equivalently $E \leftarrow C_1 \wedge \dots \wedge C_n \wedge \text{not } D_1 \wedge \dots \wedge \text{not } D_m$

the *conclusion* E is an atomic formula,

the *conditions* C_i are atomic formulas, and

the *conditions* $\text{not } D_j$ are the negations of atomic formulas.

All variables are universally quantified.

If m is 0, then the conditional is called a *definite clause*.

Goal clauses have the form: $C_1 \wedge \dots \wedge C_n \wedge \text{not } D_1 \wedge \dots \wedge \text{not } D_m$

All variables are existentially quantified.

If m is 0, then the *goal clause* is called a *definite goal clause*.

Minimal models of definite clause programs - review

In classical logic, a sentence C is a *logical consequence* of a set of sentences S if and only if C is *true in every model of S* .

Theorem: For every definite clause program P , there exists a unique minimal model M such that for all definite goal clauses G :

G is a *logical consequence* of P
(i.e. G is *true* in all models of P)
if and only if G is *true* in M .

In other words, for definite clause programs P , and definite goal clauses G , *truth in all models* and *truth in the minimal model M* are equivalent.

The *minimal model* of a definite clause program is generated by instantiating universally quantified variables with ground terms and by reasoning forwards.

Minimal models of definite clause programs - review

Let E be: $even(0)$
 $even(s(s(X))) \leftarrow even(X)$

Forward reasoning generates the infinite sequence of atomic sentences:

$even(0), even(s(s(0))), even(s(s(s(s(0))))), \dots$

This set M is the minimal model of E .

The maximal model:

$\{even(0), even(s(0)), even(s(s(0))), even(s(s(s(0))))\}, \dots\}$

is also a model of E .

Another model is

$M \cup \{even(s(s(weird))), even(s(s(s(s(weird))))), even(s(s(s(s(s(s(weird))))))\}, \dots\}$

But the Herbrand interpretation $M \cup \{even(s(0))\}$ is not a model, because it doesn't contain $even(s(s(s(0))))$.

Negation as failure for definite programs is equivalent to truth in the minimal model

Let P be: *mary will go if john will go.*
john will go if mary will go.

The minimal model of P is {},

in which no atomic sentence is *true*.

Therefore all atomic sentences are *false*.

Therefore all negations of atomic sentences are *true*.

Therefore *mary will not go.*

and *john will not go.*

This reflects the fact that the failure to show *mary will go* means that *not mary will go* is true in the minimal model of the program.

For definite clauses, the equivalence between truth in all models and truth in the minimal model does not hold for goal clauses containing negation

For example, the goal clause $\text{not even}(s(s(s(0))))$

is *true* in the minimal model M of E :

$$\begin{array}{l} \text{even}(0) \\ \text{even}(s(s(X))) \leftarrow \text{even}(X) \end{array}$$

because the atomic sentence $\text{even}(s(s(s(0))))$ is not *true* in M .

However, it is not a logical consequence of E , because it is not *true* in all models of E .

For example, it is not *true* in the maximal model of E (in which all atomic sentences are *true*).

The equivalence does not hold for goals containing universal quantification

For example, $\forall X (even(s(s(X))) \rightarrow even(X))$

is *true* in the minimal model M of E

because for any ground terms t that can be constructed from the constant 0 and the function symbol s :

if $even(s(s(t)))$ is *true* in M , then it must have been derived by forward reasoning using the ground instance $even(s(s(t))) \leftarrow even(t)$ of the conditional in E . But then the condition $even(t)$ of this ground instance must also have been derived by forward reasoning and must also be *true* in M .

But $even(s(s(t)))$ is not *true* in all models of E , because there exist non-Herbrand models containing weird individuals, for example the individual named **weird**.

such that $even(s(s(\text{weird})))$ is *true*, but $even(\text{weird})$ is not *true*.

The simplest and smallest such model is

$$M \cup \{ even(s(s(\text{weird}))), even(s(s(s(s(\text{weird}))))), even(s(s(s(s(s(s(\text{weird}))))))), ... \}$$

The distinction between operational rules and emergent properties

The clauses/beliefs: $even(0)$

$$even(s(s(X))) \leftarrow even(X)$$

are operational definitions of the even predicate
(used to generate the minimal model).

The sentences:

$$not\ even(s(s(s(0))))$$

$$\forall X (even(s(s(X))) \rightarrow even(X))$$

are emergent properties/goals

(true in the resulting minimal model).

A similar distinction holds between programs and their properties.
Failure to make the distinction is responsible for much confusion.

The stable model semantics for logic programs with negative conditions

Let P be: *mary will go if john will go.*
john will go if bob will not go.

There are two minimal models:

{bob will go}
and *{mary will go, john will go}*

{bob will go} is a model of P , because it makes the conditions of the two clauses in P false, which makes the two clauses in P true.

But negation as failure derives that *not bob will go* succeeds.
The stable model semantics for this program allows only the second minimal model, which is compatible with negation as failure.

The stable model semantics for logic programs P with negative conditions

To check that a Herbrand interpretation M is a stable model of P , let Δ be the set of all the **negative literals** $not\ q$ such that *$not\ q$ is true in M* . i.e.:

$$\Delta = \{not\ q \mid q \notin M\}$$

(A **literal** is an atomic sentence or the negation of an atomic sentence).

Generate the unique minimal model $min(P \cup \Delta)$ of $P \cup \Delta$ by treating the program $P \cup \Delta$ as a definite clause program, reasoning forwards with the negative literals $not\ q$ as though they were positive atoms.

Then M is a **stable model** of P if and only if $M = min(P \cup \Delta)$. i.e.:

If $not\ q \in \Delta$ then $q \notin min(P \cup \Delta)$
(which means that including $not\ q$ in Δ was correct).

If $q \notin min(P \cup \Delta)$ then $not\ q \in \Delta$
(which means that the inclusion of $not\ q$ in Δ was necessary for completeness).

The stable model semantics:

$$M = \text{min}(P \cup \Delta), \text{ where } \Delta = \{\text{not } q \mid q \notin M\}$$

Let P be: *mary will go if john will go.*
 john will go if bob will not go.

Candidate stable model $M = \{\text{bob will go}\}.$
 $\Delta = \{\text{not mary will go, not john will go}\}$
 M is not stable because $\text{min}(P \cup \Delta) = \{\}.$

Candidate stable model $M = \{\text{mary will go, john will go}\}.$
 $\Delta = \{\text{not bob will go}\}$
 Δ is stable because $\text{min}(P \cup \Delta) = \{\text{mary will go, john will go}\}.$

The stable model semantics is the basis for **answer set programming**, a recent, new logic programming method.

The stable model semantics:

$$M = \text{min}(P \cup \Delta), \text{ where } \Delta = \{\text{not } q \mid q \notin M\}$$

A program can have more than one stable model:

$$P: \begin{array}{l} p \leftarrow \text{not } q \\ q \leftarrow \text{not } p \end{array}$$

has two stable models $\{p\}$ and $\{q\}$.

A program can have no stable models:

$$P: p \leftarrow \text{not } p$$

Truth versus proof in arithmetic

The standard model A of arithmetic is the minimal model of the definite clause program:

$$\begin{array}{ll} +(0, Y, Y) & \text{i.e. } 0 + Y = Y \\ +(s(X), Y, s(Z)) \leftarrow +(X, Y, Z) & \text{i.e. } s(X) + Y = s(X + Y) \\ \\ \times(0, Y, 0) & \text{i.e. } 0 \times Y = 0 \\ \times(s(X), Y, V) \leftarrow \times(X, Y, U) & \\ \wedge +(U, Y, V) & \text{i.e. } s(X) \times Y = (X \times Y) + Y \end{array}$$

The Peano axioms of arithmetic are emergent properties, which are true in A .

Infinitely many instances can **sometimes** be inspected finitely by using mathematical induction

$\forall X (+(X, 0, X))$ is true in A .

Base case: $X = 0$. Then $+(X, 0, X)$ is $+(0, 0, 0)$, which is true in A

because it is an instance of the clause $+(0, Y, Y)$.

Inductive case: $X = s(n)$. By induction hypothesis, $+(n, 0, n)$ is true in A .

We need to show $+(s(n), 0, s(n))$ is true in A .

But this follows by forward reasoning, using

$+(s(X), Y, s(Z)) \leftarrow +(X, Y, Z)$.

This semantic argument can be expressed purely syntactically, by augmenting the definite clauses with axioms for induction, as in Peano arithmetic.

Truth versus proof in arithmetic

The **incompleteness** of any axioms for arithmetic is a consequence of the fact that

to show a universally quantified or negative sentence is true, it is necessary to inspect infinitely many atomic sentences, and

There exist sentences, whose infinitely many instances do not conform to any finitely recurring pattern.

These infinitely many instances **cannot be** inspected finitely even using mathematical induction.

Lecture 14

Conclusions

- Computational logic as the language of thought
- Computational logic for better communication
- Computational logic for conflict resolution
- Computational logic as a unifying framework

Computational Logic as the Language of Thought

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

It follows that:

If you want your communications to be easy to understand, then you should express them in a form that is as close as possible to the Language of Thought.

We can get insight into the nature of the Language of Thought by studying advise about effective natural language communication. (Perhaps the best book for English is [Joseph William's *Style*](#).)

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

	<i>Preface</i>	ix
1	Causes	1
2	Clarity	17
3	Cohesion	45
4	Emphasis	67
5	Coherence I	81
	<i>With Gregory G. Colomb</i>	
6	Coherence II	97
	<i>With Gregory G. Colomb</i>	
7	Concision	115
8	Length	135
9	Elegance	153
10	Usage	169
	<i>Notes</i>	199
	<i>Acknowledgments</i>	201
	<i>Index</i>	203

To express yourself clearly in natural language

1. **Avoid ambiguity.** e.g.

John gave the book to Mary.
instead of:
He gave her the book.

} clarity

2. **Avoid unnecessary complexity.** e.g.

Because we knew nothing about local conditions,
we could not determine how effectively the committee had allocated
funds to areas that most needed assistance.

Instead of:

Our lack of knowledge about local conditions precluded
determination of committee action effectiveness in fund allocation
to those areas in greatest need of assistance.

} simplicity

3. **Connect** related ideas together.

} coherence

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.

Computational logic as a model of the LOT can help people to communicate more effectively

By expressing communications:

Clearly So that their intended meaning is less ambiguous.

Simply So that their expression is closer to their mental representation.

Coherently So that it is easier for the reader/listener to connect new information to old information.

Important related topics missing from this course/book

Learning

Uncertainty

Case based reasoning

Connection with connectionism (neural networks)

Multi-agent systems

Conflict resolution

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

Achievement goals:

Improve enjoyment of life

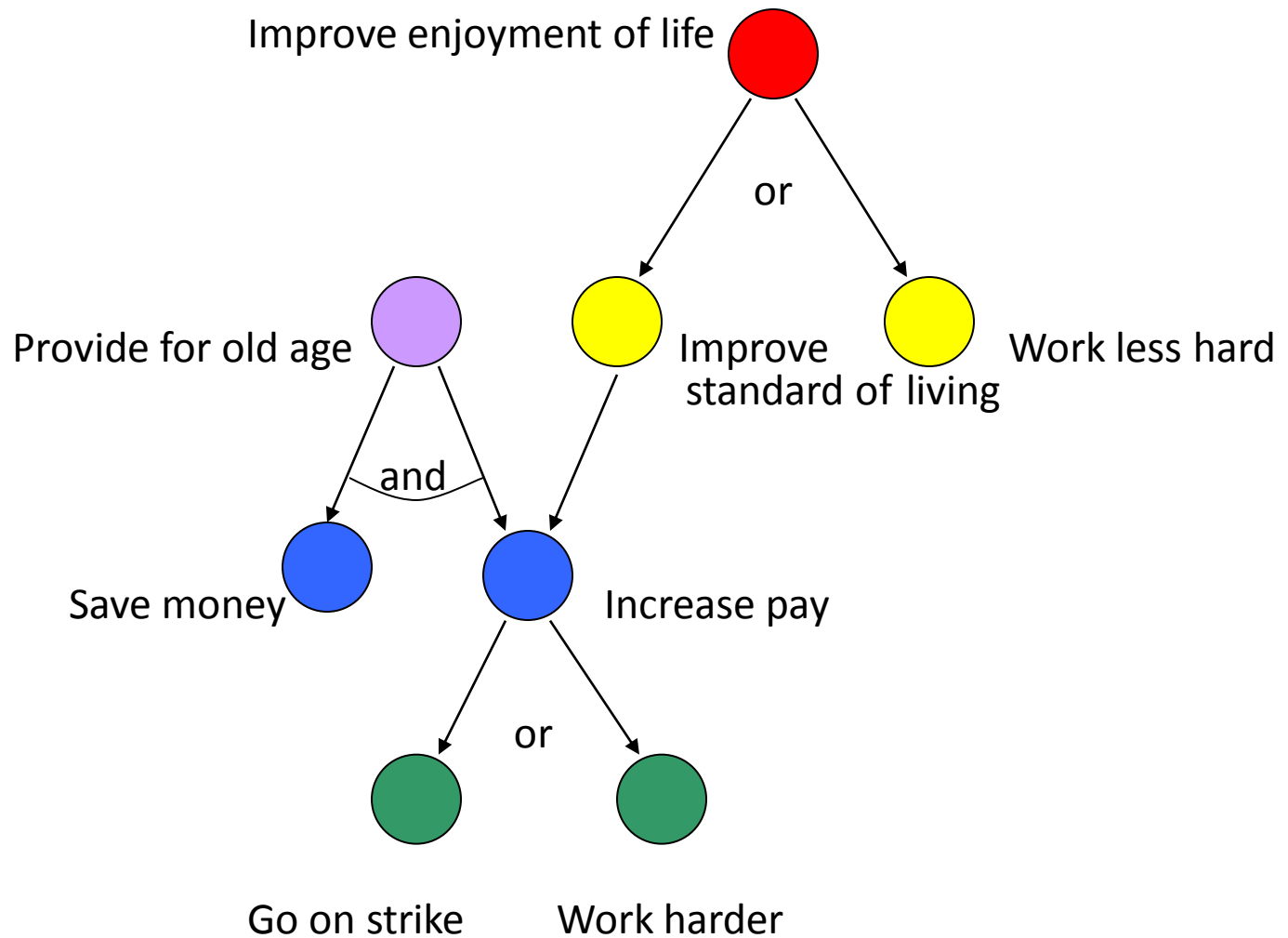
Provide for old age

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



The Israeli-Palestinian Conflict

“The Last Negotiation” by Hussein Agha and Robert Malley,
Foreign Affairs, May/June 2002

Israel's basic interests (goals)

- 1) Jewish character and majority in Israel
- 2) Security
- 3) International recognition and normalcy
- 4) Control over Jewish holy sites and national symbols
- 5) End conflict with Palestinians and Arab States.

Palestinian basic interests (goals)

- 1) Live in freedom, dignity, equality and security
- 2) End occupation and achieve national self-determination
- 3) Resolve refugee issue fairly
- 4) Control over Muslim and Christian holy sites
- 5) Ensure solution is accepted by Arab and Muslim worlds.

Proposed solution:

- 1) Territorial issue. Land swaps, with the equivalent of 100% of the land lost in 1967.
- 2) Security. Non-militarization of Palestine and international force.
- 3) Jerusalem. Demographic and religious self-governance.
- 4) Haram al-Sharif or Temple Mount. Practical arrangements to meet both sides needs.
- 5) Palestinian refugees. Settle refugees in Arab-populated parts of Israel and include these in the land swap with Palestine, and also provide generous financial compensation.

Conflict analysis and resolution

To reconcile conflicting goals G_A of agent A and G_B of agent B,

it is sometimes possible to find higher-level goals H_{A1} of agent A and H_{B1} of agent B

and additional goals H_{A2} of agent A and H_{B2} of agent B

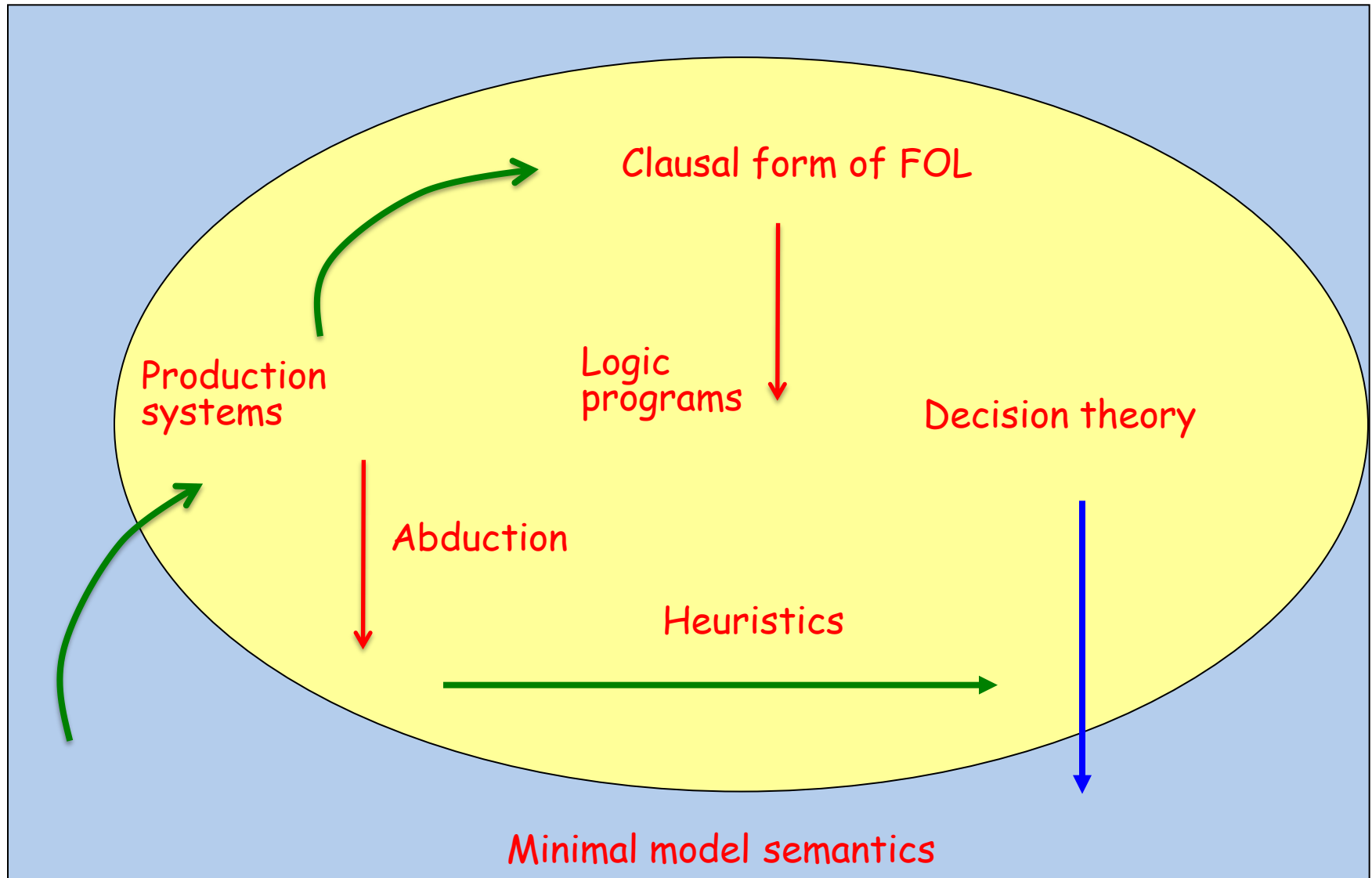
and solve the co-operative goal

H_{A1} and H_{A2} and H_{B1} and H_{B2}

subject to some minimum degree of satisfaction of both

H_{A1} and H_{A2} and H_{B1} and H_{B2}

Computational Logic as a unifying framework



Computational Logic as a unifying framework

