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)

+ Abductive Logic Programming Agents with Destructive Databases (with Fariba Sadri) an extended version of An Agent Language with Destructive Assignment and Model-theoretic Semantics, In CLIMA XI - Computational Logic in Multi- Agent Systems (eds. J. Dix, G. Governatori, W. Jamroga and J. Leite) Springer, 2010.

Computational Logic and Human Thinking

AI tools and techniques can

- be reconciled and be combined
- improve those of existing academic disciplines
- be used by ordinary people

Computational Logic and Human Thinking –

Overview

Psychology of Logic – Part 1

lecture 1

The Language of Thought (LOT)

lecture 2

Semantics

lecture 3

Abductive Logic Programming (ALP) proof procedure

Psychology of Logic – Part 2

ALP agents and BDI agents compared

lecture 4

Computational Logic and Human Thinking - Overview

- The Abductive Logic Programming (ALP) agent model

  - ✓ The agent cycle
  - ✓ Model-theoretic and operational semantics
  - ✓ Deciding between alternatives

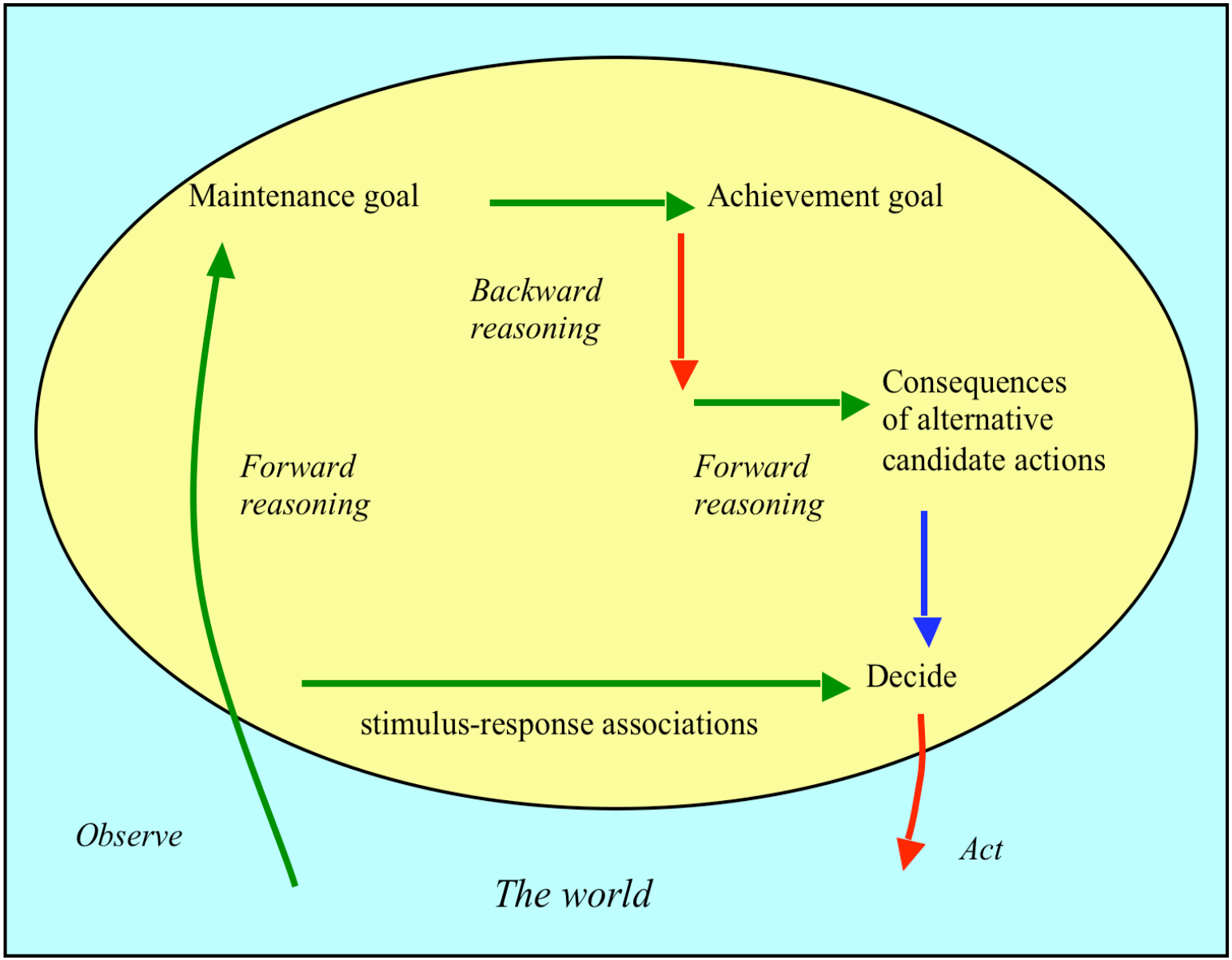
- ALP agent model as a foundation for a better decision theory

The agent cycle

repeatedly:

observe
think
decide
act

Abductive Logic Programming (ALP) Agents



ALP agent model as a dual process theory

In dual process theories, intuitive thinking “quickly proposes intuitive answers to judgement problems as they arise”, while deliberative thinking “monitors the quality of these proposals, which it may endorse, correct, or override”.

Kahneman and Frederick [2002]

The ALP agent model as a variant of BDI agents

Agents use their *beliefs* to attain their *desires* by generating *intentions*, which are selected plans of actions.

In ALP agents, beliefs and desires (or goals) are both represented as conditionals in the clausal form of logic.

Beliefs are represented as logic programming clauses, whereas goals are represented as range-restricted clauses in full first-order logic (FOL).

Goals and Beliefs

Goals: (reason forwards)

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

Beliefs: (reason forwards or backwards)

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.

The syntax of goals and beliefs

Beliefs: Logic programming *clauses* have the form:

conclusion if $condition_1$ *and* $condition_2$ …. *and* $condition_n$

where *conclusion* is an atomic formula and
$condition_i$ are atomic formulas or negations of atomic formulas.

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

If all $condition_i$ are atomic formulas, then the clause is a *Horn clause*.

All variables are universally quantified with scope the entire clause.

If every variable occurs in an unnegated condition,
then the clause is *range-restricted*.

The syntax of goals and beliefs

Goals: include clauses of the form:

If $condition_1$ *and* $condition_2$ *.... and* $condition_n$
then $conclusion_1$ *or* $conclusion_2$ *.... or* $conclusion_m$

where $condition_i$ and $conclusion_i$ are atomic formulas.
All variables in a clause are universally quantified.
A clause *is range-restricted* if every variable in *conclusion* occurs in *conditions*.

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

it is not the case that
$condition_1$ *and* $condition_2$ *.... and* $condition_n$

ALP 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.

More formally:
Given beliefs *B*, goals *G* and observations *O,*
the task 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 $B \cup \Delta$.

If *B* is a set of Horn clauses, then $B \cup \Delta$ has a unique minimal model.
Other cases can be reduced to the Horn clause case.

ALP agents - operational semantics

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

Derive the corresponding instance of the conclusion of the goal as an *achievement goal*, to make *true*.

> This is like forward chaining in production systems, but it has the semantics of aiming to make the goal *true* by making its conclusion *true* whenever its conditions become *true*.
>
> Conditional goals understood in this way are called *maintenance goals*.

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

> Executable actions are a special case of atomic sub-goals.

Operational semantics - example

Observe: *there is a fire*.

Forward reasoning: *there is an emergency*.

Forward reasoning, derive achievement goal:
I deal with it myself or I get help or I escape.

Backward reasoning: reducing
the goal *I get help*
to the consecutive sub-goals
I alert the driver of the train and
I press the alarm button.

If this last sub-goal is an atomic action
and the action is executed successfully,
then the action makes the achievement goal and
this instance of the maintenance goal both *true*.

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*.

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

Observations treated as goals

In ALP agents, observations *O* and goals *G* are treated similarly.

Given beliefs *B,* goals *G* and observations *O,*
generate actions and other assumptions Δ
to make $G \cup O$ *true*
in the minimal model of the world determined by $B \cup \Delta$.

Given *B* and *G* above and *O* = {*there is smoke*},
then $B \cup \Delta$ makes *G* and *O* both true.
where Δ = {*there is a fire, I press the alarm button*}

The operational semantics is sound and
(with sufficient restrictions) complete
with respect to the model-theoretic semantics.

Choosing the Best Solution

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.

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

Combining searching and deciding

In ALP agents, finding the best Δ is incorporated into
the search strategy for generating Δ.

This can be performed using some form of best-first search,
like A* or branch-and-bound.
This task is analogous to conflict resolution in production systems.

Conventional production systems avoid complex decision-theory
and abductive reasoning by compiling higher-level goals, beliefs and decisions
into lower-level heuristic rules. For example:

if there is smoke and I am on a train
then I press the alarm button.

In ALP agents, lower-level rules and higher-level thinking and deciding can be
combined, as in dual process theories, to get the best of both worlds.

Interleaving thinking, observing and acting

Like BDI agents, ALP agents interleave thinking with observing and acting, and do not need to construct complete plans before starting to act.

Unlike most BDI agents, which select and commit to a single plan at a time, ALP agents select and commit only to individual actions.

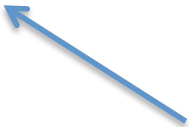
Unlike most BDI agents, ALP agents can interleave the pursuit of several alternative plans, to improve the chances of success.

For example, in an emergency an agent can both press the alarm button and try to escape at more or less the same time.

ALP reconciles logic and probability by associating probability with conditions

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 [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.

The ALP agent model provides a simple framework for making Smart Choices

Decision making between alternatives is integrated with generation of the alternatives.

The same criteria of expected utility, which is used in classical decision theory to choose between alternatives, can also be used to guide the search for alternatives in some form of best-first search.

Moreover, the ALP agent model shows how heuristics and stimulus-responses can be combined with logical thinking and decision theory in the spirit of dual process models.

Conclusions

ALP 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 make smarter choices.

Computational Logic and Human Thinking – Psychology – Part 1

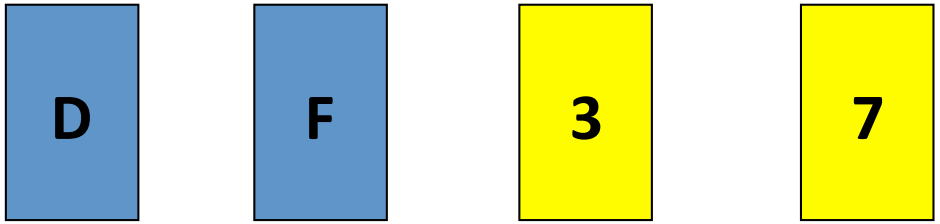
Robert Kowalski
Imperial College London

The Psychology of Logic - Part 1

- The Wason selection task and its variants
- The suppression task
- Reasoning with cause and effect

Wason selection task

Four cards, letters on one side, numbers on the other.

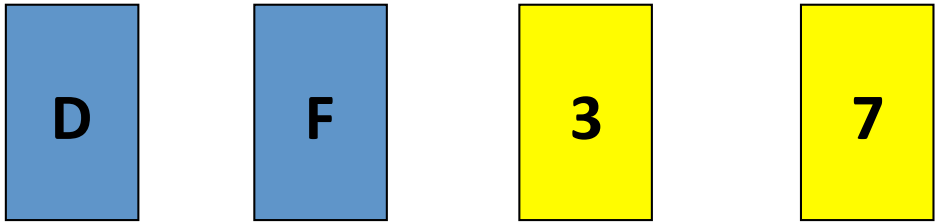


Determine whether the following rule holds:

If D is on one side,
then 3 is on the other side.

Only 5-10% of all people select the right cards.

Wason selection task – two “mistakes”



If D is on one side,
then 3 is on the other side.

Most people unnecessarily turn over the card showing 3.
(reasoning with the *converse*)

Most people fail to turn over the card showing 7.
(failing to reason with the *contra-positive*)

The selection task

Determine whether the following rule holds:

> If a person is drinking beer in a bar,
> then the person should be over eighteen.

Most people get the right answer.

Conclusion (Cosmides, 1985, 1989):
People don't use logic, but have evolved a cheater detection scheme:

> If you receive a benefit,
> you must meet its requirement.

Alternative conclusion (Cheng and Holyoak, 1985):
Subjects interpret belief (or descriptive) conditionals
and goal (or deontic) conditionals differently.

An ALP explanation of the selection task

People interpret :

- Belief conditionals as logic programming clauses:

  All the conditionals with the same conclusion are the *only* conditionals with that conclusion
  (which justifies reasoning with the converse).

  Almost all reasoning is backwards or forwards
  (which inhibits reasoning with contra-positives).

- Goal conditionals as FOL clauses
  (which explains reasoning in accordance with classical logic).

Specialized algorithm = specialized knowledge + general-purpose reasoning.

The cheater detection algorithm can be viewed as combining a specialized goal (or constraint) of the logical form:

if a person accepts a benefit
and the person fails to meet its requirement
then false.

with general-purpose reasoning of the form:

Given a conditional goal *if conditions then conclusion.*

Reason forward to match an observation with a *condition* of the goal.
Reason backward to verify the *other conditions* of the goal.
Reason forward to derive the *conclusion* as an achievement goal.
But *false* can never be achieved.

The Psychology of Logic - Part 1

- The Wason selection task and its variants
- The suppression task
- The representation of cause and effect

The suppression task (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:

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.*

The suppression task 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.*

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

In general, rules and exceptions have the form:

If conditions, then conclusion.
But *If other conditions, then **not** conclusion.*

Conclusion: The suppression task illustrates:
poor use of natural language,
not necessarily poor logical thinking.

Two alternative ways of representing the logic of rules and exceptions

Argumentation: An argument supporting an exception attacks an argument supporting the general rule. The attacking argument wins if there is no successful counterattack. etc.

Strict rules: The rule and the exception have implicit conditions:

*If conditions and **not** other-conclusion, then conclusion.*
*If other-conditions and **not** yet-other-conclusion, then other-conclusion.*
*If yet-other-conditions and **not** yet-yet-other-conclusion, then yet-other-conclusion.*
*If yet-yet-other-conditions and **not** yet-yet-yet-other-conclusion, then yet-yet-other-conclusion.*

The suppression task represented by strict rules

If she has an essay to write,
and she is not prevented from studying late in the library
then she will study 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 has an important meeting.

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

etc.

Rules and exceptions can be compiled into simple rules

If she has an essay to write,
and the library is open
*and she does **not** have an important meeting*
*and she is **not** distracted, **etc.***
then she will study late in the library.

In general, rules and exceptions:

If conditions, then conclusion.
but *If other conditions, then other-conclusion.*

can be compiled into:

*If conditions and **not** other conditions, then conclusion.*
If other conditions, then other-conclusion.

In natural language it is common to omit conditions

If she has an essay to write,
~~and she is not prevented from studying late in the library~~
then she will study late in the library.

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

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 a lot of other more complicated~~
~~but infrequently occurring conditions.~~

The Psychology of Logic - Part 1

- The Wason selection task and its variants
- The suppression task
- The representation of cause and effect

The use of conditionals to explain observations

Suppose I tell you that: *An object is red if it looks red.*
This apple looks red.

You will probably conclude: *This apple is red*.

Suppose I now say: *An object looks red*
if it is illuminated by a red light.

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

The philosopher John Pollock (1995) explains the example as an illustration of argumentation: The new information supports an argument that defeats the original argument supporting the original conclusion *This apple is red*.

An alternative explanation

The example confuses operation rules and emergent properties and

The operation rules are: ~~*An object is red if it looks red.*~~

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

An emergent property is:

An object is red ***or*** *the object is illuminated by a red light if it looks red.*

i.e. *An object is red if it looks red* ~~*and the object is* ***not*** *illuminated by a red light*~~

Conclusions

- The selection task and its variants

  - fails to distinguish between belief conditionals and goal conditionals
  - confuses natural language conditionals with logical conditionals
  - confuses the relationship between specialised algorithms and general-purpose reasoning
  + draws attention to the problems of reasoning with negation

- The suppression task

  - confuses natural language conditionals with logical conditionals
  - employs a poor representation of rules and exceptions
  + draws attention to the fact that natural language often suppresses hidden conditions

- The red light example

  - combines two alternative representations of cause and effect
  + illustrates a potential relationship between argumentation and rules and exceptions

Computational Logic and Human Thinking – The Language of Thought (LOT)

Robert Kowalski
Imperial College London

Computational Logic and Human Thinking - LOT

- The clausal logic of ALP as the LOT

  ✓The London Underground Emergency Notice
  ✓Coherence

- The relationship between natural language and the LOT ≈ the relationship between FOL and ALP

- ALP clausal logic as a connectionist model of the mind

ALP as the LOT

In ALP agents, clausal logic serves as an agent's private language of thought, independently of any
public language for communicating with other agents.

In the philosophy of language, there are three schools of thought regarding the relationship between private and public languages:

> The LOT is a private, language-like representation, which is independent of public, natural language.
>
> The LOT is a form of public, natural language; and the natural languages that we speak influence the way we think.
>
> Human thinking does not have a language-like structure at all.

ALP as the LOT

Clear, coherent communications, which are easy to understand, can help identify the LOT.

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

One of the consequences of relevance theory is that,
if you want your communications to be easy to understand,
then you should express them in a form
that is close to the meaning that you want to convey.

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 is a goal-reduction procedure, whose logic is a logic programming clause:

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

The second sentence is explicitly in logic programming clausal form, but is ambiguous; and one of its conditions has been omitted. Its obvious intended meaning is:

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 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 relative clause *where help can more easily be given* adds an extra conclusion to the sentence rather than an extra condition.

If the relative clause were meant to add an extra condition, then this would mean that the driver will not necessarily stop the train at the next station, but at the next station where help can more easily be given.

The Logic of the London Underground Notice

The fourth sentence is also a conditional, but in disguise:

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

The conditional is meant to be used
not backwards as a goal-reduction procedure,
but forwards to monitor and reject candidate actions.

Coherence is sometimes the enemy of clarity

In English sentences, omitting conditions and other details sometimes promotes coherence.

Williams [1990, 1995] in his guidelines for English writing style emphasizes another way of achieving coherence:

> Placing old, familiar ideas at the beginning of sentences and new ideas at their end.
>
> In a succession of sentences,
> a new idea at the end of one sentence becomes
> an old idea that can be put
> at the beginning of the next sentence.

Coherence by anchoring new information in relation to old information

It is raining.

If it is raining and you go out without an umbrella, then you will get wet.

If you get wet, then you may catch a cold.

If you catch a cold, then you will be sorry.

You don’t want to be sorry.

So you do not want to go out without an umbrella.

Computational Logic and Human Thinking - LOT

- ALP as the LOT

- The relationship between natural language and the LOT

  ≈ the relationship between FOL and ALP

  ✓The relationship between natural language and the LOT:
  Eliminate ambiguity + convert to canonical form

  ✓The relationship between FOL and the clausal form of ALP

- ALP clausal logic as a connectionist model of the mind

Natural Language and the LOT

The first problem in understanding natural language communications is to identify their intended meaning. For example, to understand “he gave her the book” it is necessary to identify the individuals referred to by “he” and “her”.

The second problem is to represent the intended meaning in a canonical form, so that equivalent communications are represented in the same way. For example, the following English sentences all have the same meaning:

John gave Mary the book.
John gave the book to Mary.
Mary received the book from John.
The book was given to Mary by John.

Natural Language and canonical form

The use of a canonical form in a mental representation makes it easier to use the representation later.

In this case, the common meaning of the different sentences could be represented either in the logical form *give(john, mary, book)* or in the more precise form:

event(e1000)
agent(e1000, john)
object(e1000, book21)
act(e1000, giving)
recipient(e1000, mary)
isa(book21, book)

The meaning of natural language sentences is determined in large part by the relationship between conditions and conclusions

The ambiguous English sentence:

"Every bird which belongs to class aves has feathers. "

means either:

every bird has feathers.
every bird belongs to class aves.

or *a bird has feathers if the bird belongs to class aves.*

The relationship between Standard FOL and Clausal Logic

Clausal logic is as powerful as standard FOL, but much simpler.

It compensates for the lack of explicit existential quantifiers by Skolemization to give individuals that are supposed to exist a name,
like the names *e1000* and *book21*.

Reasoning is also much simpler, and for the most part can be reduced to just forward and backward reasoning, which are both special cases of the resolution rule

Standard FOL is to clausal form as natural language is to the LOT.

In both cases, inferences can be partitioned into two kinds, performed in two stages.

The first kind converts sentences into canonical form, and the second kind reasons with that canonical form.

In FOL, the first kind of inference rule (including both Skolemization and the replacement of *not(A or B)* by *not A and not B*) can be viewed as converting sentences into clausal form.

The second kind (including the inference of *P(t)* from *∀XP(X))* can be viewed as reasoning with clausal form, and is built into the resolution rule.

Clausal logic is a canonical form of logic.

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

can-fly(X) ← bird(X).
bird(john).

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

¬(∃X((¬can-fly(X) ∧ bird(X)) ∨ ¬bird(john)))
¬(∃X((¬can-fly(X) ∨ ¬bird(john)) ∧ (bird(X) ∨ ¬bird(john))))

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

By expressing information:

Clearly	So that it has its meaning is unambiguous.
Simply	So that it is meaning is close to its canonical form.
Coherently	So that it is easy to link new information to old information.

Computational Logic and Human Thinking - LOT

- ALP as the LOT

- The relationship between natural language and the LOT
  ≈ the relationship between FOL and ALP

- ALP clausal logic as a connectionist model of the mind

  ✓External links ground thoughts in reality

  ✓Internal links organise thoughts, and need not have any "meaning"

  ✓Links can be activated by highest expected utility

A Connectionist Clausal Logic

In the same way that clausal logic implements FOL by reasoning in advance, the connection graph proof procedure implements clausal logic, by pre-computing links between the conditions and conclusions of clauses.

It labels links with their unifying substitutions.
These links can then be activated later, either forwards or backwards, 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 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].

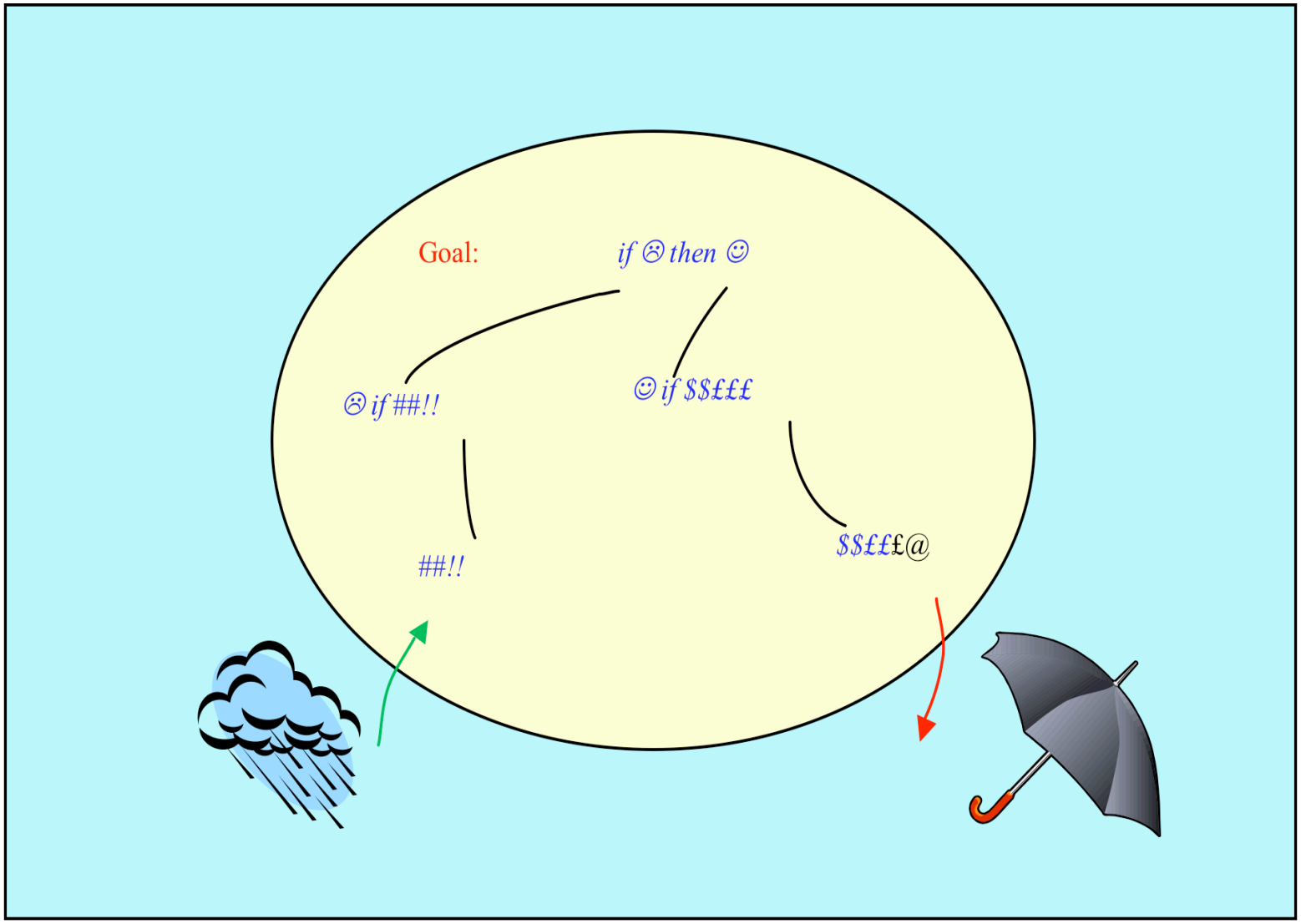
Connection graphs, the LOT and reality

Although clausal form is a symbolic representation, once all the links have been computed, the names of the predicate symbols no longer matter.

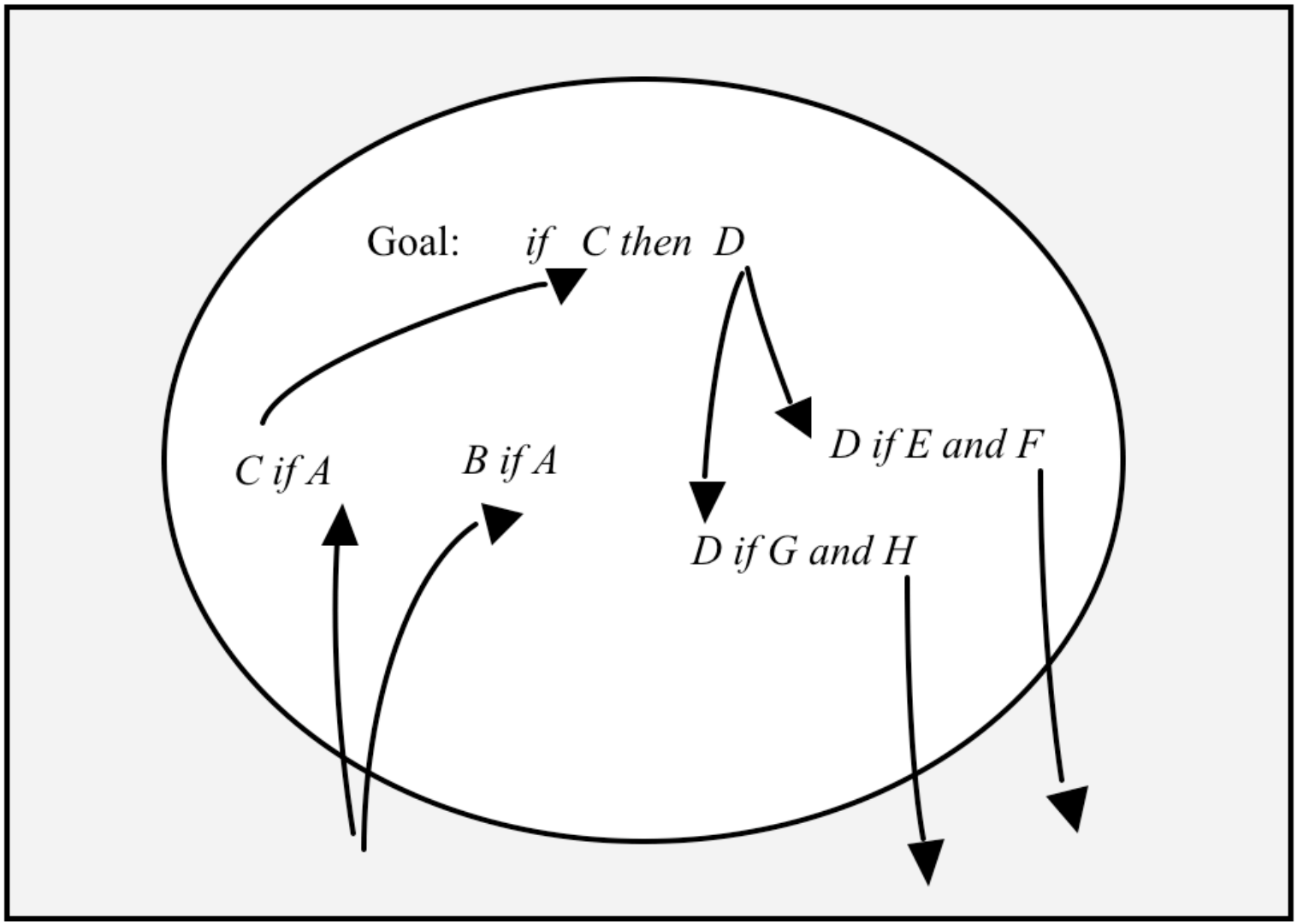
The links in connection graphs include internal links, which organize the agent's thoughts, and external links, which ground the agent's thoughts in reality.

The external links are activated by observations and by the agent's own actions. They may also include links to unobserved properties of the world.

Connection graphs and the LOT



Internal clauses and links need not represent states of affairs in the real world



Conclusion - The clausal logic of ALP as a model of the LOT

- English sentences that are closer to clausal logic
  are generally easier to understand

- English sentences that are stuctured by means of
  connections between conditions and conclusions
  are generally easier to understand

- Understanding natural language and reasoning in FOL
  similarly involve two stages of converting into canonical form
  and reasoning with canonical form

- The connection graph implementation of clausal logic

  + precomputes as much reasoning as possible in advance
  + can use weights on links and strengths of goals and observations
    to guide the activation of connections

Computational Logic and Human Thinking – Semantics

Robert Kowalski
Imperial College London

Computational Logic and Human Thinking – Semantics

- Logical consequence and truth
- An argument for Herbrand interpretations
- The paradoxes of material Implication
- Paraconsistency
- The argument for Minimal Herbrand Models
  - ✓ Minimal models of Horn clauses
  - ✓ The distinction between operational rules and emergent properties
  - ✓ Default reasoning
  - ✓ Truth versus proof in arithmetic

Logical consequence in classical logic

A sentence *C* is a *logical consequence* of a set of sentences *S*
(or *S logically implies C*)
if *C* is *true* in an interpretation
whenever *S* is *true* in the same interpretation.

A set of inference rules is *sound* (or *truth-preserving*)
if *C* is a *logical consequence* of *S*
whenever there exists a derivation of *C* from *S*.

A set of inference rules is *complete*
if there exists a derivation of *C* from *S*
whenever *C* is a *logical consequence* of *S.*
.

Truth is relative to an interpretation

An *interpretation* is a collection of *individuals* and *relations*.
(For simplicity, properties of individuals are also regarded as relations.)

In the language, constants and other variable-free (*ground*) terms
denote (or *name* or *represent*) individuals.
Predicate symbols *denote* (or *name* or *represent*) relations.
Terms can be constructed using function symbols.

The simplest way to represent an interpretation in symbolic form is
by the set of all the atomic sentences that are *true* in the interpretation.
Such interpretations are called *Herbrand interpretations*.

The truth of more complicated sentences can be reduced to
the truth of atomic sentences by the standard definition.

Example

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

then the conditional *amazing(john)* ← *can-fly(john)*

means: My cat is lazy if my cat sleeps all day.

Because my cat sleeps all day and my cat is lazy,
the sentences *can-fly(john)* and *amazing(john)* are both *true*,
the conditional is also *true*.

The definition of truth

A sentence that is not *true* is *false*.
A negative sentence $\neg C$ is *true* if C is *false*.

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

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

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

A sentence that is a conditional $C \rightarrow D$ is *true* in an interpretation
if C is *false* or D is *true.* (Therefore if C is *false,* then $C \rightarrow$ *false* is *true* .)

A universally quantified sentence $\forall X\ C$ is *true*
if every *ground instance* of C is *true*.

An existentially quantified sentence $\exists X\ C$ is *true*
if some ground instance of C is *true*.

Computational Logic and Human Thinking – Semantics

- Logical consequence and truth
- An argument for Herbrand interpretations
- The paradoxes of material Implication
- Paraconsistency
- The argument for Minimal Herbrand Models
  - ✓ Minimal models of Horn clauses
  - ✓ The distinction between operational rules and emergent properties
  - ✓ Default reasoning
  - ✓ Truth versus proof in arithmetic

Universal quantifiers, negation and Herbrand interpretations

The semantics *∀X C* is *true* if every *ground instance* of *C* is *true* is called the *substitution interpretation of quantifiers*.

For this to work, there need to be enough ground terms
to name all the individuals in the interpretation.
Not only the individuals in the set of sentences,
but any other individuals that might need talking about in the future.

The substitution interpretation does away with the mystery of what counts as an individual and what counts as a relation.
It allows interpretations to be restricted to Herbrand interpretations.

The semantics ¬ *C* is *true* if *C* fails to be *true* reflects
the asymmetry between truth and falsity.
In the ALP agent model, this asymmetry is reflected in the fact that an agent's "passive" observations are positive atomic sentences.

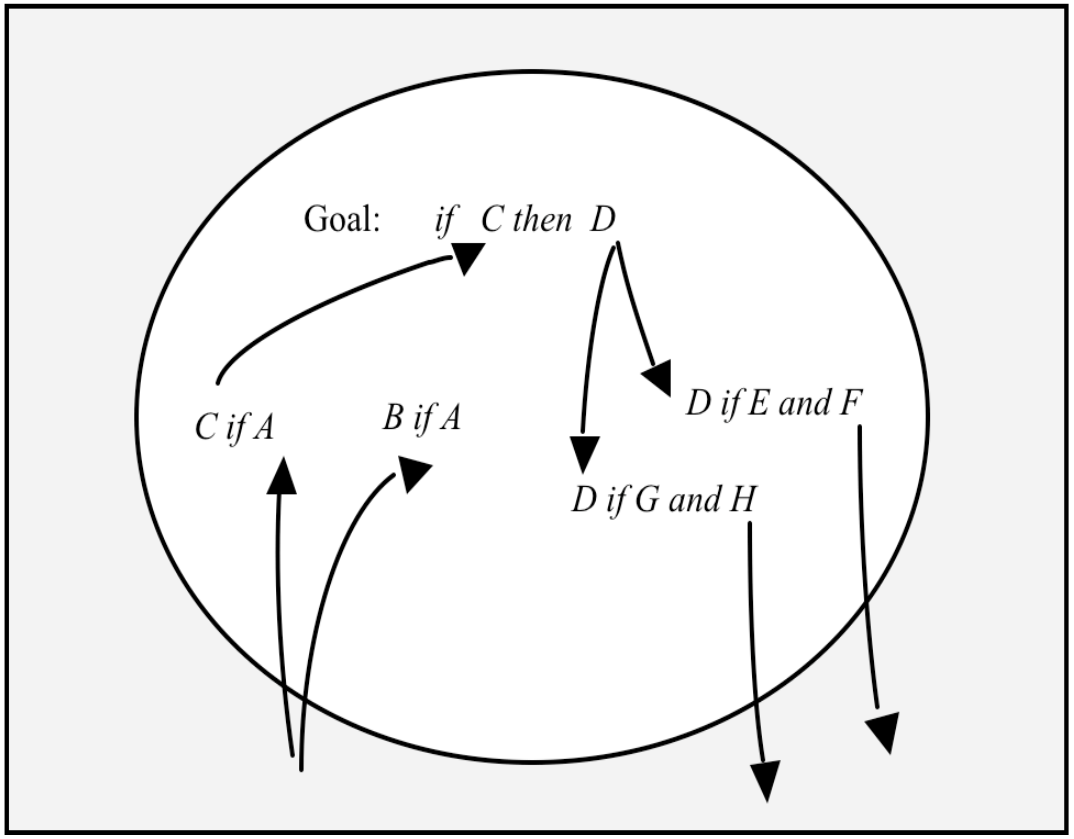
The relationship between an agent's thoughts and the Real World.

The interface between the Real World and an agent's goals and beliefs is the set of observations that the agent encounters and the set of actions that the agent performs.

This interface is as close as the agent needs to get to the Real World, to determine whether its beliefs are *true* and whether its goals can be made *true*.

The use of Herbrand interpretations restricts the agent's knowledge of the world to this interface,
and avoids trying to identify the true nature of the World without describing it in some other language.

Internal clauses and links need not represent states of affairs in the Real World



Only A, F and H correspond directly to reality.

C and D are mental constructs, which help the agent organise its thoughts.

The definition of truth *projects* these mental constructs onto the external world. This is another argument for Herbrand interpretations.

Computational Logic and Human Thinking – Semantics

- Logical consequence and truth
- An argument for Herbrand interpretations
- The paradoxes of material Implication
- Paraconsistency
- The argument for Minimal Herbrand Models
  - ✓ Minimal models of Horn clauses
  - ✓ The distinction between operational rules and emergent properties
  - ✓ Default reasoning
  - ✓ Truth versus proof in arithmetic

The paradoxes of material implication

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

The conditional is *true* whenever the conclusion D is *true*,
no matter whether the condition C is *true* or *false*.

The conditional is also *true* whenever the condition C is *false*,
no matter whether the conclusion D is *true* or *false*.

For example: *john can fly* → *2 + 2 = 4*
the moon is made from green cheese → *john can fly*

are both *true* if *2 + 2 = 4* is *true* and
the moon is made from green cheese is *false*.

But: *john can fly* → *I am a monkey's uncle*

The paradoxes are avoided in the pragmatics

Why assert the *weak disjunction*, even if it is *true*:

I am going to the party ***or*** *I will stay at home*

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

In clausal logic, the paradoxes are avoided by eliminating weak disjunctions and weak conditionals for the sake of efficiency.

The *subsumption rule* eliminates $C \vee D$
given a stronger disjunction D.

It eliminates $B \wedge C \rightarrow D$ or $C \rightarrow D \vee E$
given a stronger conditional $C \rightarrow D$.

Therefore it eliminates $B \rightarrow D$ given D
and it eliminates or $C \rightarrow E$ given $C \rightarrow \mathit{false}$ (i.e. $\neg C$)

Computational Logic and Human Thinking – Semantics

- Logical consequence and truth
- An argument for Herbrand interpretations
- The paradoxes of material Implication
- Paraconsistency
- The argument for Minimal Herbrand Models
  - ✓ Minimal models of Horn clauses
  - ✓ The distinction between operational rules and emergent properties
  - ✓ Default reasoning
  - ✓ Truth versus proof in arithmetic

Paraconsistency

The paradoxes of material implication are closely related to the property of classical logic that an inconsistent set of sentences logically implies every sentence.

Given the definition of logical consequence:

> 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*.

If *S* is inconsistent, then it is *false* that *S* is *true* in any interpretation.
Therefore *C* is *true* whenever *S* is *true*.
Therefore *C* is a logical consequence of *S*.

But it would be more informative to say:

> Given that *C* is a *logical consequence* of *S* and that *S* is inconsistent, it is impossible to say whether or not *C* is *true* in any interpretation.

Clausal logic is inconsistency-tolerant (paraconsistent)

The resolution rule derives only
informative consequences of a set of clauses.

Given only the clauses *p* and *not p*,
only one application of resolution is possible,
and it derives *false* in one step.

It doesn't derive that *the moon is made of green cheese*,
or that *the world is coming to an end*.

Computational Logic and Human Thinking – Semantics

- Logical consequence and truth
- An argument for Herbrand interpretations
- The paradoxes of material Implication
- Paraconsistency
- The argument for Minimal Herbrand Models
  - ✓ Minimal models of Horn clauses
  - ✓ The distinction between operational rules and emergent properties
  - ✓ Default reasoning
  - ✓ Truth versus proof in arithmetic

Minimal models of Horn clause programs

Every Horn clause program has a unique *minimal model*. It is the Herbrand model generated by instantiating universally quantified variables with ground terms and by reasoning forwards.

Consider the recursive definite clauses *E*:

even(0)
even(s(s(X))) ← even(X)

Forward reasoning generates the infinite set of atomic sentences:

even(0), even(s(s(0))), even(s(s(s(s(0))))),.....ad infinitum.

This set is the smallest Herbrand model that makes *E true*.

The minimal model of a Horn clause program *P*
is contained in every other Herbrand model of *P*.

For Horn clauses, truth in the minimal model is equivalent to truth in all models

For every Horn clause program *P*,
there exists a unique minimal model *M*
such that for all Horn 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*.

This is a direct consequence of a theorem in (van Emden and Kowalski, 1976) for the case where *G* is an atomic fact.
It also holds for disjunctions of definite goal clauses,
i.e. sentences of the form $G_1 \vee \ldots \vee G_n$
where each G_i is an (existentially quantified) Horn goal clause.

However, the equivalence does not hold for
goals containing negation or universal quantification.

The equivalence does not hold for goals containing negation

not even(s(s(s(0))))

is *true* in the minimal model *M* of *E*:

even(0)
even(s(s(X))) ← even(X)

because the atomic sentence *even(s(s(s(0))))* is not *true* in *M*.

However, it is not a logical consequence of *E*,
because it is not *true*, for example, in the maximal model of *E*
(in which all atomic sentences are *true*)

The equivalence does not hold for goals containing universal quantification

$$\forall X\ (even(s(s(X))) \rightarrow even(X))$$

is *true* in the minimal model *M* of *E*
because for all 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))) ← even(t)* of the conditional in *E*. But then the condition *even(t)* of this ground instance must also be *true* in *M.*

But it 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(𝔴𝔢𝔦𝔯𝔡)))* is *true*, but *even(𝔴𝔢𝔦𝔯𝔡)* is not *true*. The simplest and smallest such model is just the minimal model augmented with the one additional atomic sentence *even(s(s(𝔴𝔢𝔦𝔯𝔡)))*.

Computational Logic and Human Thinking – Semantics

- Logical consequence and truth
- An argument for Herbrand interpretations
- The paradoxes of material Implication
- Paraconsistency
- The argument for Minimal Herbrand Models
  - ✓ Minimal models of Horn clauses
  - ✓ The distinction between operational rules and emergent properties
  - ✓ Default reasoning
  - ✓ Truth versus proof in arithmetic

The distinction between operational rules and emergent properties

The clauses/beliefs: *even(0)*
even(s(s(X))) ← even(X)

are operational definitions of the even predicate.

The sentences: *not even(s(s(s(0))))*
∀X (even(s(s(X))) → even(X))

are emergent properties/goals.

A similar distinction holds between programs and their properties.
Failure to make the distinction is responsible for much confusion in psychological and philosophical studies of logic.

Computational Logic and Human Thinking – Semantics

- Logical consequence and truth
- An argument for Herbrand interpretations
- The paradoxes of material Implication
- Paraconsistency
- The argument for Minimal Herbrand Models
  - ✓ Minimal models of Horn clauses
  - ✓ The distinction between operational rules and emergent properties
  - ✓ Default reasoning
  - ✓ Truth versus proof in arithmetic

Default reasoning can be understood in terms of truth in minimal models

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 *mary will not go*.
Similarly *john will not go, as far as I know.*

Default reasoning can be understood in terms of truth in minimal models

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*}

But default reasoning and negation as failure
give higher priority to the second minimal model,

treating *john will go if bob will not go.*
as an operational rule
and the contra-positive *bob will go if john will not go.*
as an emergent property.

Two ways to Represent Cause and Effect

In the form *effect if cause:*

there is smoke if there is a fire.
there is smoke if there is teargas.

In the form *cause if effect:*

there may be a fire if there is smoke.
there may be teargas if there is smoke.

i.e. *there is a fire or there is teargas if there is smoke.*

Arguably, the *effect if cause* representation is operational,
whereas the *cause if effect* representation is emergent
(in the minimal model).

The *effect if cause* representation needs abduction to explain observations.
The *cause if effect* representation needs only deduction.

Computational Logic and Human Thinking – Semantics

- Logical consequence and truth
- An argument for Herbrand interpretations
- The paradoxes of material Implication
- Paraconsistency
- The argument for Minimal Herbrand Models
  - ✓ Minimal models of Horn clauses
  - ✓ The distinction between operational rules and emergent properties
  - ✓ Default reasoning
  - ✓ Truth versus proof in arithmetic

Truth versus proof in arithmetic

The standard model *A* of arithmetic is
the minimal model of the Horn clauses:

+(0, Y, Y) i.e. *0 + Y =Y*

+(s(X), Y, s(Z)) ← +(X, Y, Z) i.e. *s(X) + Y = s(X + Y)*

×(0, Y, 0) i.e. *0 × Y =0*

×(s(X), Y, V) ← ×(X, Y, U) ∧ +(U, Y, V) i.e. *s(X) × Y = (X × Y) + Y*

The Peano axioms are emergent properties,
which are true in *A*.

Infinitely many instances can sometimes by inspected finitely by using mathematical induction

$\forall X\ (+(X, 0, X))$ is *true* in A.

Base case: $X = 0$. Then $+(X, 0, X)$ is just $+(0, 0, 0)$, 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 constructive 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
>
> these infinitely many instances
> do not conform to any finitely recurring pattern.

The same incompleteness applies to
default reasoning and properties of logic programs.

Computational Logic and Human Thinking – Semantics - Conclusions

- Logical consequence and truth
- An argument for Herbrand interpretations
- The paradoxes of material Implication
- Paraconsistency
- The argument for Minimal Herbrand Models
  - ✓ Minimal models of Horn clauses
  - ✓ The distinction between operational rules and emergent properties
  - ✓ Default reasoning
  - ✓ Truth versus proof in arithmetic

Computational Logic and Human Thinking – Semantics - Conclusions

- The definition of truth in classical logic applies to the clausal logic of ALP
- Herbrand interpretations avoid the problem of trying to identify the individuals and relations in arbitrary interpretations
- The paradoxes of material Implication can be resolved in the pragmatics of classical logic
- Paraconsistency can be achieved in the pragmatics of classical logic
- Truth in Minimal Herbrand Models
  - ✓ is equvalent to logical consequence for Horn clause programs
  - ✓ is more fundamental than logical consequence
  - ✓ distinguishes between operational rules and emergent properties
  - ✓ is an appropriate semantics for default reasoning
  - ✓ explains the distinction between truth and proof in arithmetic

Computational Logic and Human Thinking

The Logic of Abductive Logic Programming

Robert Kowalski
Imperial College London

The Logic of Abductive Logic Programming

- Semantics
- Abductive derivations
- Stable models
- Admissibility semantics
- Argumentation-theoretic interpretation

ALP - Abstract semantics

An *abductive logic program* <*P, O, IC*> consists of a logic program *P*, a set of open predicates *O* and a set of integrity constraints *IC.*

The open predicates are restricted so they do not occur in the conclusions of clauses in *P*.

A *solution* of a goal clause *G* is a set *Δ* of ground instances of the open predicates *O* such that:

> *G* holds with respect to the program *P U Δ* and
> *P U Δ* satisfies *IC*.

G can be an observation to explain or an achievement goal.
IC can be maintenance goals and constraints.

ALP - Concrete semantics

A *solution* of a goal clause *G*
is a set *Δ* of ground instances of the open predicates *O* such that:

G is true in the minimal model of *P U Δ* and
IC is true in the minimal model of *P U Δ.*

i.e. *{G} U IC* is true in the minimal model of *P U Δ.*

The notion of *minimal model* is clear-cut in the case in which
P U Δ is a Horn clause program.
This case is the basis for all other cases and extensions.

The Logic of Abductive Logic Programming

- Semantics
- Abductive derivations
- Stable models
- Admissibility semantics
- Argumentation-theoretic interpretation

An inference system for ground Horn ALP

A *ground Horn* abductive logic program <*P, O, IC*>
consists of a ground (variable-free) Horn clause program *P*,
a set of open predicates *O,* and integrity constraints *IC,*
which are ground conditionals of the form:

$$A \wedge B \rightarrow C.$$

where *A* is an *open atom*
(i.e. an atom with an open predicate in *O*),
and *B* and *C* are conjunctions of atoms.

Integrity constraints of this form are like
event-condition-action rules of active databases
(Widom and Ceri, 1996).

The problem is to solve a ground Horn goal clause G_0,
which is a conjunction of variable-free atoms.

Abductive derivation

The ALP proof procedure is adapted from the IFF proof procedure for ALP (Fung and Kowalski, 1997). But the IFF proof procedure uses logic programs expressed in the biconditional *if and only if* form and employs the theoremhood view of integrity satisfaction.

The ALP proof procedure uses forward and backward reasoning to generate a solution Δ of G_0
by generating an abductive derivation $G_0, G_1, \ldots G_N$
such that G_N contains the set Δ
but no other goals that need to be solved.

G_{i+1} is obtained from G_i by one of:

F_1 *Forward reasoning* with a selected open atom A in $G_i = A \wedge G$ and an integrity constraint $A \wedge B \rightarrow C$. Then $G_{i+1} = (B \rightarrow C) \wedge A \wedge G$.
(The resulting goal clause is a *generalised goal clause*.)

F_2: *Forward reasoning* with a selected open atom A and a conditional in $G_i = (A \wedge B \rightarrow C) \wedge A \wedge G$. Then $G_{i+1} = (B \rightarrow C) \wedge A \wedge G$.

B_1: *Backward reasoning* with a selected atom C in $G_i = C \wedge G$ and a clause $C \leftarrow D$ in P. Then G_{i+1} is $D \wedge G$.

B_2: *Backward reasoning* with a selected atom C in $G_i = (C \wedge B \rightarrow H) \wedge G$.
Suppose $C \leftarrow D_1$ ..... $C \leftarrow D_m$ are all the clauses in P having conclusion C.
Then $G_{i+1} = (D_1 \wedge B \rightarrow H) \wedge .... \wedge (D_m \wedge B \rightarrow H) \wedge G$.

Fact: *Factoring* $G_i = A \wedge A \wedge G$. Then $G_{i+1} = A \wedge G$.
(Any previous applications of F_1 and F_2 to any occurrence of A are deemed to have been done to the resulting single copy of A.)

S: *Logical simplification:*
Replace *true* $\rightarrow C$ by C.
Replace *true* $\wedge C$ by C.
Replace *false* $\wedge C$ by *false*.

Successfully terminating derivation

An abductive derivation $G_0, G_1, \ldots G_N$ is a *successfully terminating derivation* of $\Delta = \{ A_1, \ldots, A_n \}$ if and only if:

G_N is not *false*

$G_N = (B_1 \rightarrow C_1) \wedge \ldots \wedge (B_m \rightarrow C_m) \wedge A_1 \wedge \ldots \wedge A_n$, $m \geq 0$, $n \geq 0$ where

each A_i is an open atom and

no further inferences can be performed on G_N

The residual conditionals $B_i \rightarrow C_i$ are conditionals introduced by F_1 but whose conditions B_i are not *true* in the minimal model of $P \cup \Delta$.

The inference rules are *sound*

Theorem: Given a ground Horn abductive logic program $\langle P, O, IC \rangle$ and ground Horn goal clause G_0:

> If there exists a successfully terminating derivation of Δ then $\{G_0\} \cup IC$ is *true* in the minimal model of $P \cup \Delta$.

The inference rules are not complete, because they do not recognise infinite failure.

Infinite success and incompleteness

Given < *{C ← C}, {A}, {A ∧ C → false}*> and the goal *A,* the inference rules generate the non-terminating derivation:

G_0	*A*		given
G_1	*(C → false) ∧ A*		by F_1
G_2	*(C → false) ∧ A*		by B_2
	ad infinitum	……	by B_2

This infinite derivation is the only derivation possible.

However, *Δ = {A}* is a solution of G_0
because the integrity constraint and the initial goal
are both true in the minimal model of *P ∪ {A}*.
The integrity constraint *A ∧ C → false* is *true*, because *C* is *false*.

Non-constructive completeness can be obtained by broadening the definition

An abductive derivation $G_0, G_1, \ldots G_N$ is a *successful derivation* of $\Delta = \{A_1, \ldots, A_n\}$ if and only if:

G_N is not *false*

$G_N = (B_1 \rightarrow C_1) \wedge \ldots \wedge (B_m \rightarrow C_m) \wedge A_1 \wedge \ldots \wedge A_n$, $m \geq 0$, $n \geq 0$ where

each A_i is an open atom
no further inferences can be performed on the A_i and
the conditions B_i of the residues are not *true*
in the minimal model of $P \cup \Delta$.

Implementing the last requirement can be done by trying to show that the B_i are *true* and failing. This is impossible in general, but can be achieved in many cases by *tabling* (Sagonas, Swift and Warren, 1994).

Theorem: Given a ground Horn abductive logic program <*P, O, IC*>,
a ground Horn goal clause G_0 and a set of ground open atoms Δ:

If $\{G_0\} \cup IC$ is *true* in the minimal model of $P \cup \Delta$,
then there exists a successful derivation of Δ', such that $\Delta' \subseteq \Delta$.

Integrity constraints with disjunctive conclusions

To deal with integrity constraints of the form:

$$C \rightarrow D_1 \vee \ldots \vee D_m$$

it suffices to add the inference rule:

Splitting: If G_i has the form $(D_1 \vee \ldots \vee D_m) \wedge G$,
then there are as many successor nodes G_{i+1}
of the form $D_i \wedge G$ as there are disjuncts D_i.

Splitting needs to be performed when the conditions of an integrity constraint are reduced to *true*, and the disjunctive conclusion is added to G_i.

The splitting rule, together with F_1 and F_2, turns the ALP proof procedure into a model generator for (range-restricted) clausal logic.

The case <*P, O, IC*> where *P* is empty and *O* is the set of all predicates in the language is equivalent to SATCHMO (Manthey and Bry, 1988).

The Logic of Abductive Logic Programming

- Semantics
- Abductive derivations
- Stable models
- Admissibility semantics
- Argumentation-theoretic interpretation

The stable model semantics

is the special case of the ALP minimal model semantics in which

negations of atoms *not a* are treated as
positive, open atoms, say *non-a*
and integrity constraints express that *a* and *non-a* are contraries:

Consistency constraints: $\textit{non-a} \wedge \textit{a} \rightarrow \textit{false}$

Totality constraints: $\textit{true} \rightarrow \textit{non-a} \vee \textit{a}$

The correspondence with stable models

For every logic program with negation *P,*
there is a *corresponding* abductive logic program <*P', O, IC*> where

O is the set of positive contraries *non-a* of the negations of atoms in *P*,
P' is the Horn clause program obtained from *P*
by replacing negative conditions *not a* with *non-a* , and
IC is the set of consistency and totality constraints.

With this correspondence the stable models of *P*
coincide with the minimal models of *P' ∪ Δ*,
where *Δ* is a solution of the initial goal *true* (Eshghi and Kowalski, 1989).

However, there is a problem with the correspondence:
It requires the satisfaction of all the totality constraints
whether they are relevant to the initial goal G_0 or not.

The case for ignoring the totality constraints

Consider the program *P*: *bob will go ← not john will go.*
john will go ← not bob will go.

P': *bob will go ← john stays away.*
john will go ← bob stays away.
O: *{john stays away, bob stays away}*
IC: *bob will go ∧ bob stays away → false.*
john will go ∧ john stays away → false.

G_0 *bob will go*
G_1 *john stays away*
G_2 *(john will go → false) ∧ john stays away*
G_3 *(bob stays away → false) ∧ john stays away*

The proof procedure generates only one successfully terminating derivation with solution Δ_1 = *{john stays away}*:
The result is the same as that obtained with the stable model semantics, but without totality constraints.

The case for the totality constraints

Consider the program *P*: *john can fly* ← *john is a bird* ∧ *not(john is abnormal)*
john is a bird

In most semantics for default reasoning, it can be shown that
john can fly but not that *not(john can fly).*

But without totality constraints it can be shown that *john is flightless*, where:

P'	*john can fly* ← *john is a bird* ∧ *john is normal*
	john is a bird
O	*{john is flightless, john is normal}*
IC:	*john is flightless* ∧ *john can fly* → *false.*
	john is normal ∧ *john is abnormal* → *false.*
G_0	*john is flightless*
G_1	*(john can fly* → *false)* ∧ *john is flightless*
G_2	*(john is a bird* ∧ *john is normal* → *false)* ∧ *john is flightless*
G_3	*(john is normal* → *false)* ∧ *john is flightless*

This is also a counter-example to replacing totality constraints by the requirement that $P' \cup \Delta$ be maximally consistent.

An alternative to the totality constraints

The effect of restricting the totality constraints to those that are locally relevant to the goal can be obtained by adding the inference rules:

Neg: If G_i has the form $(non\text{-}C \wedge B \rightarrow H) \wedge G$
then G_{i+1} is $(B \rightarrow H \vee C) \wedge G$.

Replace $non\text{-}C \wedge C$ by *false*
Replace *false* $\vee$ C by C.

An alternative to the totality constraints

P'	*john can fly ← john is a bird ∧ john is normal*
	john is a bird
O	*{john is flightless, john is normal}*
IC:	*john is flightless ∧ john can fly → false.*
	john is normal ∧ john is abnormal → false.

The first three steps are the same as they were before without the totality constraint:

G_0	*john is flightless*
G_1	*(john can fly → false) ∧ john is flightless*
G_2	*(john is a bird ∧ john is normal → false) ∧ john is flightless*
G_3	*(john is normal → false) ∧ john is flightless*

Before the derivation terminated successfully with G_3.
Now negation rewriting applies, and the derivation terminates unsuccessfully with G_4:

G_4	*john is abnormal ∧ john is flightless*

The derivation terminates unsuccessfully,
because the subgoal *john is abnormal* is not an open atom,
and no further inferences can be applied.

Preventative maintenance

The combination of *Neg* and *Splitting* makes it possible to satisfy maintenance goals by preventing the need to achieve their conclusions

P: *you fail the exam ← you do not study.*
O: *{you have an exam, you study,*
you do not study, you retake the exam}
IC: *you have an exam ∧ you fail the exam → you retake the exam.*
you study ∧ you do not study → false.

G_0 *you have an exam*
G_1 *you have an exam ∧ (you fail the exam → you retake the exam)*
G_2 *you have an exam ∧ (you do not study → you retake the exam)*
G_3 *you have an exam ∧ (you study ∨ you retake the exam)*
G_4 *you have an exam ∧ you study*
G_4' *you have an exam ∧ you retake the exam*

The Logic of Abductive Logic Programming

- Semantics
- Abductive derivations
- Stable models
- Admissibility semantics
- Argumentation-theoretic interpretation

An argumentation-theoretic interpretation

An abductive derivation using *Neg* can be viewed as constructing an argument to support and defend an initial claim G_0:

B_1 reduces the initial goal to an argument supported by open subgoals *non-a* added to Δ.

When an open atom *non-a* is added to Δ, F_1 is used with the consistency constraint to derive $a \rightarrow$ *false*, in an attempt to attack *non-a*.

B_2 reduces *a* in $a \rightarrow$ *false* to alternative arguments attacking *non-a.* Each such attacking argument is ultimately reduced to a conjunction of open subgoals of the form *non-b*.

For each such attacking argument, Neg attempts to defeat the attack by finding a *non-b* in the attack and showing *b*.

In a successful derivation, this dialectic process continues until every attack against the open atoms in Δ has been counter-attacked.

An argumentation-theoretic semantics

Δ is an *admissible* solution of a goal G_0 if and only if:

- $P' \cup \Delta$ supports an argument for G_0.
- No argument supported by $P' \cup \Delta$ attacks Δ.
- For every argument supported by $P' \cup \Delta'$ that attacks Δ, $P' \cup \Delta$ supports an argument that attacks Δ'.

In the admissibility semantics
argumentation is self-defence.

An argumentation-theoretic interpretation of the stable model semantics

Given an abductive logic program <*P', O, IC*> corresponding to a normal logic program *P*, the stable model semantics can sanctions a set *Δ* of open atoms as a solution of a goal G_0 if and only if:

- *P' U Δ* supports an argument for G_0.
- No argument supported by *P' U Δ* attacks *Δ.*
- For every *non-b* not in *Δ,*
  P' U Δ supports an argument that attacks *non-b.*

In the stable model semantics
argumentation is all-out warfare:
Every *non-b* has to take a side, either with or against *Δ*.
If *non-b* is not with *Δ*, then *Δ* attacks *non-b.*

Extensions of the abductive proof procedure

The extension to non-ground abductive logic programs
requires mainly just adding unification.
It also requires the range-restriction on variables.

Forward reasoning needs to be generalized, so that the atom A in G_i used for forward reasoning can be a closed atom. This allows the consequences of hypothetical actions and explanations to be considered without the need to reduce them to open atoms.

Logic programs need to be generalized to include conditionals in the conditions.

Forward reasoning needs to be generalized, to reason forwards using beliefs, in addition to integrity constraints. This involves relaxing the restriction that every integrity constraint contains an atom with an open predicate.

The abductive proof procedure needs to be intergrated with the connection graph proof procedure. i.e. adding, inheriting and deleting links, subsumption, etc.

The Logic of Abductive Logic Programming – Conclusions

- Semantics in terms of model generation
- Includes the SATCHMO theorem-prover,
  i.e range-restricted clauses of FOL
- Includes stable model semantics,
  admissibility semantics
- Has an argumentation-theoretic interpretation

Computational Logic and Human Thinking – Psychology Part 2

Robert Kowalski
Imperial College London

Computational Logic and Human Thinking – Psychology Part 2

- The conditional in the selection task interpreted as a belief

- The conditional in the selection task interpreted as a goal

An abstract form of the selection task

Assume that an agent has been told that a sentence having the logical form:

if P then Q.

ought to be true, but might be false.

Assume, moreover, that *P* and *Q* are open predicates that are directly observable. The abstract form of the selection task is to determine how the agent should respond to various observations of the truth values of these predicates.

The abstract form of the selection task

According to classical logic, the correct responses are:

From an observation of *P* deduce *Q.* *(modus ponens)*
From an observation of *not Q* deduce *not P.* *(modus tollens)*

However, in some variants of the selection task, including the original card version, most people:

From an observation of *P* deduce *Q.* *(modus ponens)*
From an observation of *Q* deduce *P.*
(affirmation of the consequent)

The ALP explanation of the selection task

People interpret :

- Belief conditionals as logic programming clauses:

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

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

- Goal conditionals as FOL clauses
  (which explains reasoning in accordance with classical logic).

A more accurate representation of the selection task

if X has value u for property p then X has value v for property q.

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

if a person X is drinking alcohol in a bar
then the person X has age at least eighteen years old.

In many cases, the values *u* and *v* are a function of X. For example:

if X has value V for property q and X has value W for property q
then V is identical to W.

where *is identical to* is defined by: *X is identical to X*.

For example: *if a card X has number N on the number side*
and the card X has number M on the number side
then N is identical to M.

Computational Logic and Human Thinking – Psychology Part 2

- The conditional in the selection task interpreted as a belief

- The conditional in the selection task interpreted as a goal

The conditional interpreted as a belief.

Modus Ponens is easy.

Affirmation of the Consequent is natural.

Modus Tollens is hard, but possible.

Denial of the Antecedent is hard and hardly ever applied.

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:

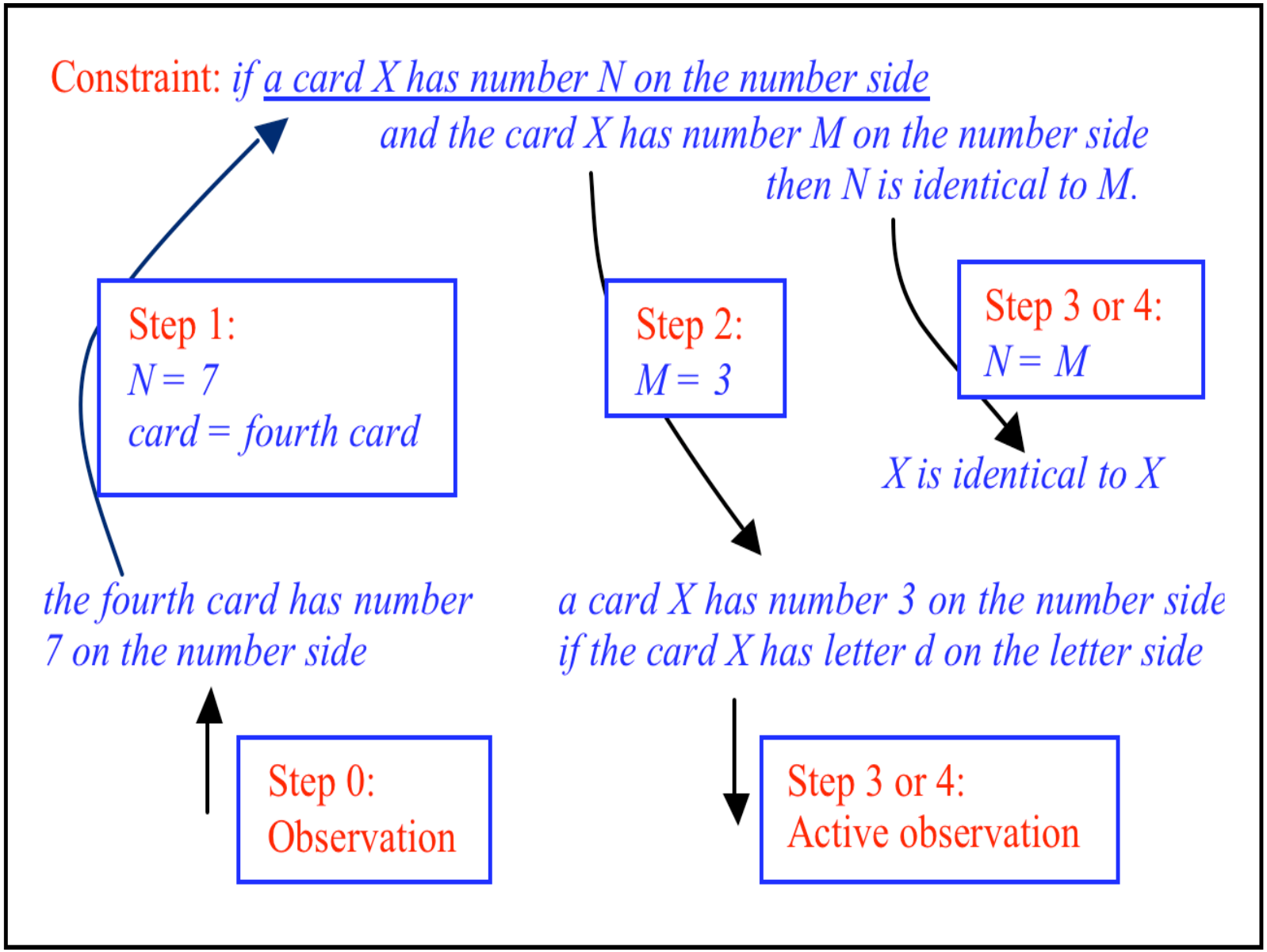
it is not the case that the fourth card has number 3 on the number side.

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 conditional interpreted as a belief – *Modus Tollens is hard*



Computational Logic and Human Thinking – Psychology Part 2

- The conditional in the selection task interpreted as a belief

- The conditional in the selection task interpreted as a goal

The conditional interpreted as a goal.

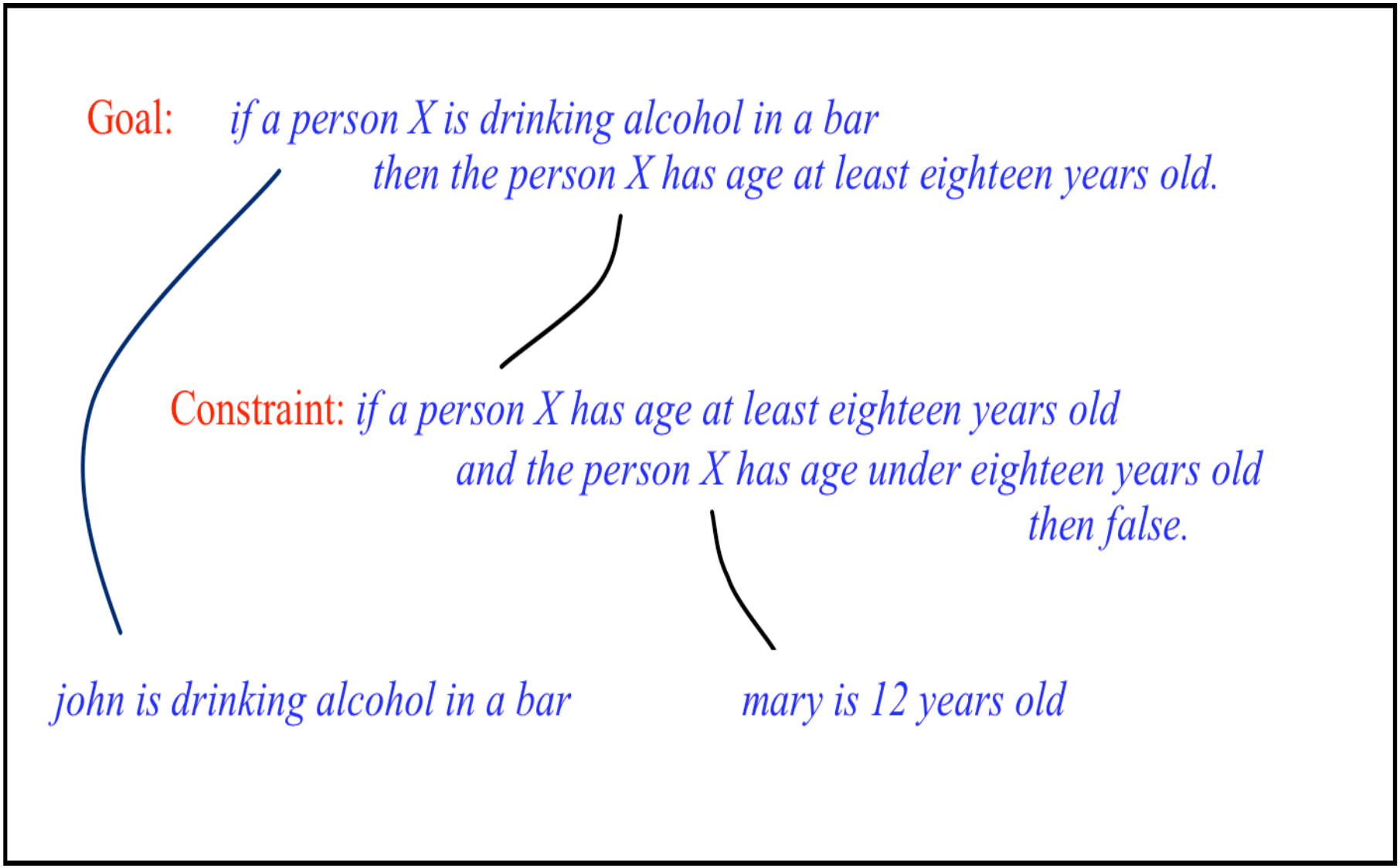
Modus Ponens is easy.

Affirmation of the Consequent has no justification.

Modus Tollens is potentially easier.

Denial of the Antecedent has no justification

The conditional interpreted as a goal –
Modus Tollens is easier, because the link between the goal and the constraint is stronger.



In general, if the conditional is interpreted as a goal,
then it is natural to compile the goal and constraint
into a more specialized goal

Conditional goal: *if P then Q.*

Constraint: *if Q and Q' then false.*

Compiled goal: *if P and Q' then false.*
Or equivalently: *it is not the case that P and Q'.*

This assumption is similar to the argument of (Sperber *et al*, 1995), that subjects are likely to perform *modus tollens*, if they interpret the conditional *if P then Q* as a denial:

i.e. *it is not the case that P and not Q.*
or equivalently *if P and not Q then false.*

Computational Logic and Human Thinking – Psychology Part 2
Conclusions

- The meaning of natural language conditionals depends on whether they are interpreted as beliefs or as goals.

- Modus tollens is hard, partly because deriving negative conclusions from positive observations is hard.

- We need to reason with the equivalence between goals of the form:

if P and not Q then R.

and *if P then R or Q.*

But it is not clear when this reasoning should be performed.

Computational Logic and Human Thinking – ALP agents and BDI agents compared

Robert Kowalski (and Fariba Sadri)
Imperial College London

Computational Logic and Human Thinking – ALP agents and BDI agents compared

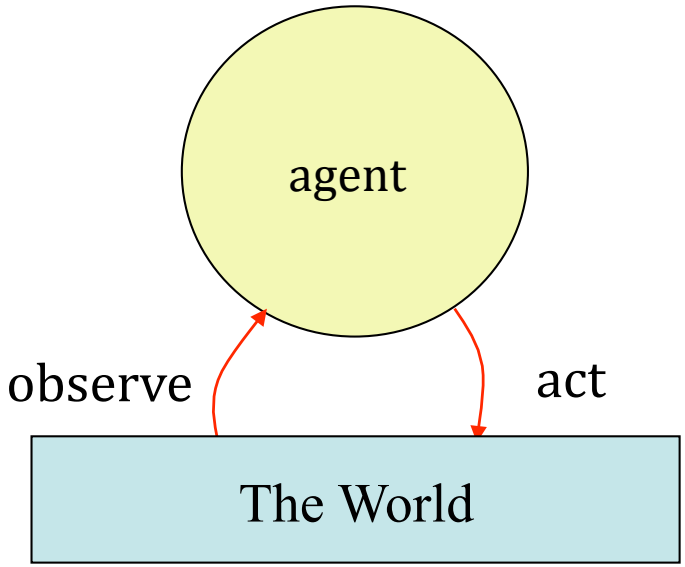
- Three kinds of rules
- Robot room-cleaning example – BDI vs ALP/LPS
- Teleo-reactivity
- LPS syntax and model-theoretic semantics
- LPS operational semantics
- Soundness and Incompleteness

ALP agents and BDI agents compared- The agent cycle

Production systems, BDI agents and ALP agents share a similar agent cycle:

repeatedly:

observe
think
decide
act



But in ALP agents the world gives meaning (semantics) to the agent's thoughts.

ALP agents and BDI agents compared- Knowledge representation

Production systems and typical BDI agents
"represent knowledge" by means of a

destructively updated database (or working memory),
which represents the current state of the world

and

rules (in production systems)
if condition then action(s) or

plans (in BDI agents)
(*trigger, guard, body*) i.e.
If triggering condition and guard conditions
hold in the current state of the database,
then execute body (solving goals and performing actions).

In ALP agents, the database is non-destructive, and observations or frame axioms are need to access the current state of the world.

Computational Logic and Human Thinking – ALP agents and BDI agents compared

- Three kinds of rules
- Robot room-cleaning example – BDI vs ALP/LPS
- Teleo-reactivity
- LPS syntax and model-theoretic semantics
- LPS operational semantics
- Soundness and Incompleteness

Production systems and BDI agents – three kinds of rules/plans

Reactive rules, e.g.

If there is an emergency then get help.
If there is an emergency then run away.

may have implicit/emergent goals, e.g.

deal with the emergency.

In ALP, these are represented by maintenance goals/ integrity constraints, but expressed declaratively, e.g.

If there is an emergency then I deal with the emergency
I deal with the emergency if I get help
I deal with the emergency if I run away

"equivalently" *if there is an emergency then I get help or I run away*
not *if there is an emergency then I get help and I run away*

Production systems and BDI agents – three kinds of rules/plans

Goal-reduction rules:

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

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

from *Introduction to Cognitive Science*, (Thagard, 2005) page 45.

In ALP: *G if C and H*

e.g. *You go home*
if you have the bus fare and you catch a bus
(and maybe other missing/implicit conditions).

Production systems and BDI agents – three kinds of rules/plans

Forward reasoning rules, e.g.

If X mother of Y
*then **add** X ancestor of Y.*

If X ancestor of Y
and Y ancestor of Z
*then **add** X ancestor of Z.*

In ALP:

If X mother of Y
then X ancestor of Y.

If X ancestor of Y
and Y ancestor of Z
then X ancestor of Z.

Computational Logic and Human Thinking – ALP agents and BDI agents compared

- Three kinds of rules
- Robot room-cleaning example – BDI vs ALP/LPS
- Teleo-reactivity
- LPS syntax and model-theoretic semantics
- LPS operational semantics
- Soundness and Incompleteness

Robot example from Dennis, L.A. Farwer, B. Bordini, R.H. Fisher, M. Wooldridge, M. A. (2008). Common Semantic Basis for BDI Languages LICS 4908, Springer-Verlag, 124-139.

PLAN 1: trigger +!aclean()
prefix [ε]
guard stack dirty(Room)
TRUE
body +!aGoto(Room)
+!aVacuum(Room)

PLAN 2: trigger+!Goto(R)
prefix [ε]
guard stack pos(P)
TRUE
TRUE
body -pos(P)
+pos(R)
+Goto(R)

Robot example in LPS (Logic-based Production System) - ALP agents with stateless syntax and destructive database

Reactive rule (or maintenance goal) in ***R***:

observe-dirty(Room) → *make-clean(Room)*

Definitions of intensional predicates (beliefs) in ***I***:

clean(Room) ← ¬ *dirty(Room).*

Macro-action definitions (beliefs) in ***Mac***:

make-clean(Room) ← *clean(Room).*
make-clean(Room) ← *dirty(Room)* **:** *goto(Room); vacuum(Room).*
goto(Y) ← *pos(Y).*
goto(Y) ← ¬ *pos(Y)* **:** *pos(X)* **:** *adjacent(X, Z): enter(X, Z)* **;** *goto(Y).*

: means "immediately afterwards" **;** means "afterwards"
dirty, *pos*, *clean* and *adjacent* are database predicates
make-clean is a macro-action, *enter* and *vacuum* are atomic actions
observe-dirty is an observation predicate

LPS internal syntax

Reactive rule in ***R****:

observe-dirty(Room, T-1, T) → make-clean(Room, T1, T2) ∧ T ≤ T1

i. e. *∀ Room, T-1, T(observe-dirty(Room, T-1, T)*
→ ∃ T1, T2(make-clean(Room, T1, T2) ∧ T ≤ T1))

Intensional predicates in ***I****: *clean(Room) ← ¬ dirty(Room).*

Macro-action definitions in ***Mac****:

make-clean(Room, T, T) ← clean(Room, T).
make-clean(Room, T1, T2) ← dirty(Room, T1) ∧ goto(Room, T1, T)
∧ T ≤ T' ∧ do(vacuum(Room), T', T2).

goto(Y, T, T) ← pos(Y, T).
goto(Y, T1, T2) ← ¬ pos(Y, T1) ∧ pos(X, T1) ∧ adjacent(X, Z)
∧ do(enter(X, Z), T1, T) ∧ T ≤ T' ∧ goto(Y, T', T2).

State transitions of the destructive database are defined by an action theory ***Act***

initiates(observe-dirty(Room), dirty(Room)).
initiates(enter(X,Y), pos(Y)).
terminates(enter(X,Y), pos(X)).
terminates(vacuum(Room), dirty(Room)).

The action theory also includes specifications of preconditions of the atomic actions:

possible(enter(X,Y)) ← pos(X) ∧ adjacent(X,Y).
possible(vacuum(Room)) ← pos(Room).

There are no frame axioms! The frame axioms are not operational rules, but emergent properties that are true in the minimal model associated with the agent's beliefs and the sequence of database states generated by the agent cycle.

Given a sequence of observations $\boldsymbol{O} = O_1, O_2, O_3, \ldots$
and initial state E_0, the LPS agent cycle generates a sequence
$E_1, E_2, E_3, \ldots$ of database states and actions $\boldsymbol{A} = A_1, A_2, A_3, \ldots$

For simplicity, observations are treated as events/actions.
So abduction is not necessary.

E_0 = *{pos(room3), adjacent(room3, room1)}*
O_1 = *{ observe-dirty(room1)}*
A_1 = ϕ (the empty set of actions)

E_1 = *{pos(room3), adjacent(room3, room1), dirty(room1)}*

The agent records the observation and its effects in the database,
reasons forwards with the reactive rule, and
generates the achievement goal *make-clean(room1)*.
It reasons backwards from the achievement goal
with the macro-action definitions, and
selects the atomic action *enter(room3, room1).*

Given a sequence of observations, the LPS cycle generates a sequence of database states and actions

If the action has been executed successfully, then it uses the action theory and any new observations to update the database:

O_2={}
A_2 = {enter(room3, room1)}
E_2 = {pos(room1), adjacent(room3, room1), dirty(room1)}

It continues to reason backwards to solve the goal *make-clean(room1)* and selects the atomic action *vacuum(room1):*

O_3={}
A_3 = {vacuum(room1)}
E_3 = {pos(room1), adjacent(room3, room1)}

It observes the successful execution of *vacuum(room1)* and deletes *dirty(room1)}* from the database. The agent has
achieved the goal *make-clean(room1)* and
maintained the goal *observe-dirty(Room)* → *make-clean(Room)*

Destructive database update and its semantics

The representation of database states without explicit state (or time) makes it easy to implement destructive updates:

Delete facts that are terminated by an action or observation
Add facts that are initiated.

To obtain a declarative semantics, transform the sequence of database states into a single database in which state is represented explicitly, e.g.:

E = { pos(room3, 0), adjacent(room3, room1), dirty(room1, 1),*
pos(room3, 1), enter(room3, room1, 1, 2), dirty(room1, 2),
pos(room1, 2), vacuum(room1, 2, 3),
pos(room1, 3)}.

The agent cycle generates a minimal model

If ***I***, ***Mac*** and ***Act*** are Horn clauses,
then ***E**** ∪ ***I**** ∪ ***O**** ∪ ***A**** ∪ ***Temp*** ∪ ***Mac*** is also a set of Horn clauses,
where ***Temp*** = *{0≤0, 0≤1, 0≤2,…1≤1, 1≤2,…..}*

Every set of Horn clauses has a unique minimal model.
This model can be represented by the set of ground atomic sentences that are *true* in the model. In this example, the minimal model is:

M = ***E**** ∪ *{observe-dirty(room1, 0, 1)}* ∪
{goto(room3, 0, 0), goto(room3, 1, 1),
goto(room1, 1, 2), goto(room1, 2, 2),
goto(room1, 3, 3), goto(room1, 1, 3),
make-clean(room1, 1, 3), make-clean(room1, 2, 3),
make-clean(room1, 3, 3)} ∪ ***Temp***

The agent cycle generates a minimal model that makes the agent's maintenance goals all *true*

The goal $\forall T\text{-}1, T(observe\text{-}dirty(Room, T\text{-}1, T) \rightarrow \exists T1, T2(make\text{-}clean(Room, T1, T2) \wedge T \leq T1))$

is *true* in the mimimal model:

$$\boldsymbol{M} = \boldsymbol{E}^* \cup \{observe\text{-}dirty(room1, 0, 1)\} \cup \{goto(room3, 0, 0), goto(room3, 1, 1), goto(room1, 1, 2), goto(room1, 2, 2), goto(room1, 3, 3), goto(room1, 1, 3), make\text{-}clean(room1, 1, 3), make\text{-}clean(room1, 2, 3), make\text{-}clean(room1, 3, 3)\} \cup \boldsymbol{Temp}$$

The minimal model semantics of Horn clauses can be generalised to the perfect model semantics of extended stratified programs.

Computational Logic and Human Thinking – ALP agents and BDI agents compared

- Three kinds of rules
- Robot room-cleaning example – BDI vs ALP/LPS
- Teleo-reactivity
- LPS syntax and model-theoretic semantics
- LPS operational semantics
- Soundness and Incompleteness

Teleo-Reactivity [Nilsson, 1994]

[AgentSpeak (BDI) goal-reduction rule in LPS:
quench-thirst ← go-to-fridge; open-fridge; get-drink; open-drink; drink.

A more flexible, teleo-reactive program in LPS:

quench-thirst ← quenched-thirst.
quench-thirst ← do-open-drink ; drink.

do-open-drink ← opened-drink.
do-open-drink ← do-get-drink ; open-drink.

do-get-drink ← have-drink.
do-get-drink ← do-open-fridge ; get-drink.

do-open-fridge ← opened-fridge.
do-open-fridge ← do-go-to-fridge ; open-fridge.

do-go-to-fridge ← at-fridge.
do- go-to-fridge ← do-stand-up : go-to-fridge.

Computational Logic and Human Thinking – ALP agents and BDI agents compared

- Three kinds of rules
- Robot room-cleaning example – BDI vs ALP/LPS
- Teleo-reactivity
- LPS syntax and model-theoretic semantics
- LPS operational semantics
- Soundness and Incompleteness

LPS syntax – the language and database

The *language* ***L*** of an LPS framework < ***I, R, Mac, Act***> consists of:

- disjoint sets of predicate symbols for:

  the extensional and intensional predicates of the deductive database ***DBL***

  macro-action predicates ***MA***

  atomic action predicates ***AA***

  observation predicates denoted by the set ***OB***

  predicates *initiates/2*, *terminates/2* and *possible/1 (used in* ***Act****)*,

- fixed set of constant symbols and function symbols.

I is a locally stratified logic program in the language of ***DBL***. ***I*** defines intensional predicates in terms of extensional and other intensional predicates in ***DBL.***

LPS syntax – queries and goal clauses

Definition: A *query to a database* D is an arbitrary FOL (first-order logic) formula (possibly containing free variables) in the vocabulary of D.

If the database is a set of atomic sentences (e.g. minimal or perfect model), then a substitution θ is an *answer* of a query d, if $d\theta$ is *true* in D.
We also say that $d\theta$ *holds* in D

Definition: Any expression of the form F where F is a query, *true* or an atomic formula representing an atomic or macro-action is a *goal clause*.

If F_1 is a query, *true* or an atomic formula representing an atomic or macro-action, and F_2 is a *goal clause*, then $F_1 : F_2$ and $F_1 ; F_2$ are also *goal clauses*.

All variables in a goal clause that are not explicitly quantified are implicitly existentially quantified with scope the entire goal clause.

Definition: F is an *extended goal clause* if it is a goal clause, or if it is of the form $: F1$ where $F1$ is a goal clause.

LPS syntax – extended stratified programs

Definition: A *reactive rule* in ***R*** is a *conditional* of the form *conditions* $\rightarrow$ *conclusion* where *conditions* is a query in the language of ***DBL*** $\cup$ ***OB*** $\cup$ ***AA*** and *conclusion* is a goal clause.

All variables that are not explicitly quantified are implicitly universally quantified with scope the entire conditional, except for variables occurring only in the *conclusion* that are existentially quantified with scope the *conclusion*. E.g. $p(X, Y) \rightarrow q(Y, Z)$ stands for $\forall Y \forall X (p(X, Y) \rightarrow \exists Z\, q(Y, Z))$.

Definition: A macro-action definition in ***Mac*** has the syntactic form *mac* $\leftarrow$ *conditions*, where *mac* is an atomic formula whose predicate symbol is a macro-action predicate, and *conditions* are a goal clause.

All variables that are not explicitly quantified are implicitly universally quantified with scope the entire clause. However, any such universally quantified variable that is not in the conclusion can be treated as existentially quantified with scope the conditions.

E.g. $p(Y) \leftarrow \forall X\, q(X, Y, Z)$ can be understood as $\forall Y (p(Y) \leftarrow \exists Z\ \forall X\, q(X, Y, Z))$.

LPS syntax – the action theory

Definition:The *action theory* ***Act*** consists of clauses of the form:

conclusion ← *conditions* where
conclusion is one of the predicates *initiates, terminates* or *possible* and
conditions is any query.

If *event* is an action that is executed successfully
or an external event is observed,
then all the extensional facts
that are terminated by *event* are deleted from the database,
and all the extensional facts
that are initiated by *event* are added to the database

The "extensions" of the intensional predicates are updated implicitly as ramifications of explicit changes to the extensional predicates.

LPS syntax is similar to that of Transaction Logic (TR logic, Bonner and Kifer), but with a different model-theoretic and operational semantics.

Extended perfect model

Definition: Let L be a locally stratified program, with perfect model D.
Let M be a program of the form: $m \leftarrow n \wedge d$
where n is a conjunction of atomic formulas, and
d is a query formula in the language of D regarded as a database.
Let the predicates of L be disjoint from the predicates in m and n.

The set of clauses M is said to be *stratified*
(at a higher level) with respect to L.

Then the *extended perfect model* of $L \cup M$
is the minimal model of the set of Horn clauses
$D \cup \{m\theta \leftarrow n\theta \mid m \leftarrow n \wedge d \in M$ and θ is an answer of the query $d\}$.

Computational Logic and Human Thinking – ALP agents and BDI agents compared

- Three kinds of rules
- Robot room-cleaning example – BDI vs ALP/LPS
- Teleo-reactivity
- LPS syntax and model-theoretic semantics
- LPS operational semantics
- Soundness and Incompleteness

Soundness.

Given an LPS framework $\langle \boldsymbol{I, R, Mac, Act} \rangle$ with a timely selection function, an initial state $\langle \boldsymbol{E_0, G_0} \rangle$ and an input stream of externally generated observations $O_0, , \ldots, O_i, , O_{i+1}$, let

$$\langle \boldsymbol{E_0, G_0} \rangle, A_0, O_0, \langle \boldsymbol{E_1, G_1} \rangle , \ldots, \langle \boldsymbol{E_i, G_i} \rangle, A_i, O_i, \ldots.$$

be generated by the operational semantics, and let ***M*** be the *extended perfect model* of $\boldsymbol{E^* \cup I^* \cup O^* \cup A^* \cup Temp \cup Mac^*.}$

If for every goal tree in every goal state $\boldsymbol{G_i}$ there exists a goal state $\boldsymbol{G_j}$, where $i \leq j$ in which some branch of that goal tree has a leaf node labelled by *true*,

then $\boldsymbol{G^*_0 \cup R^*}$ is *true* in ***M***.

The following variant of the situation calculus is an emergent property

The following axioms are all *true*
in the extended perfect model of
$\boldsymbol{E^* \cup I^* \cup Ob^* \cup A^* \cup Temp \cup Act^*}$:

$$P(S') \leftarrow initiates(E, P, S) \wedge E(S, S')$$
$$P(S') \leftarrow P(S) \wedge \neg \exists E\, (E(S, S') \wedge terminates(E, P, S))$$

where P is any intentional or extensional database predicate,
and $E(S, S')$ means that the atomic event E
takes place between the successive states S and S'.

Computational Logic and Human Thinking – ALP agents and BDI agents compared

- Three kinds of rules
- Robot room-cleaning example – BDI vs ALP/LPS
- Teleo-reactivity
- LPS syntax and model-theoretic semantics
- LPS operational semantics
- Soundness and Incompleteness

The LPS operational semantics - goal states G_i

E_0:	$\{q, s\}$	$I = \{\}$	$Act = \{\}$	$O = \{\}$
G_0 :	p			
R:	$q \rightarrow m.$		Rule 1	
	$a1 \rightarrow n.$		Rule 2	
Mac:	$p \leftarrow a1.$		Defn 1	
	$m \leftarrow r : a2.$		Defn 2	
	$m \leftarrow s.$		Defn 3	

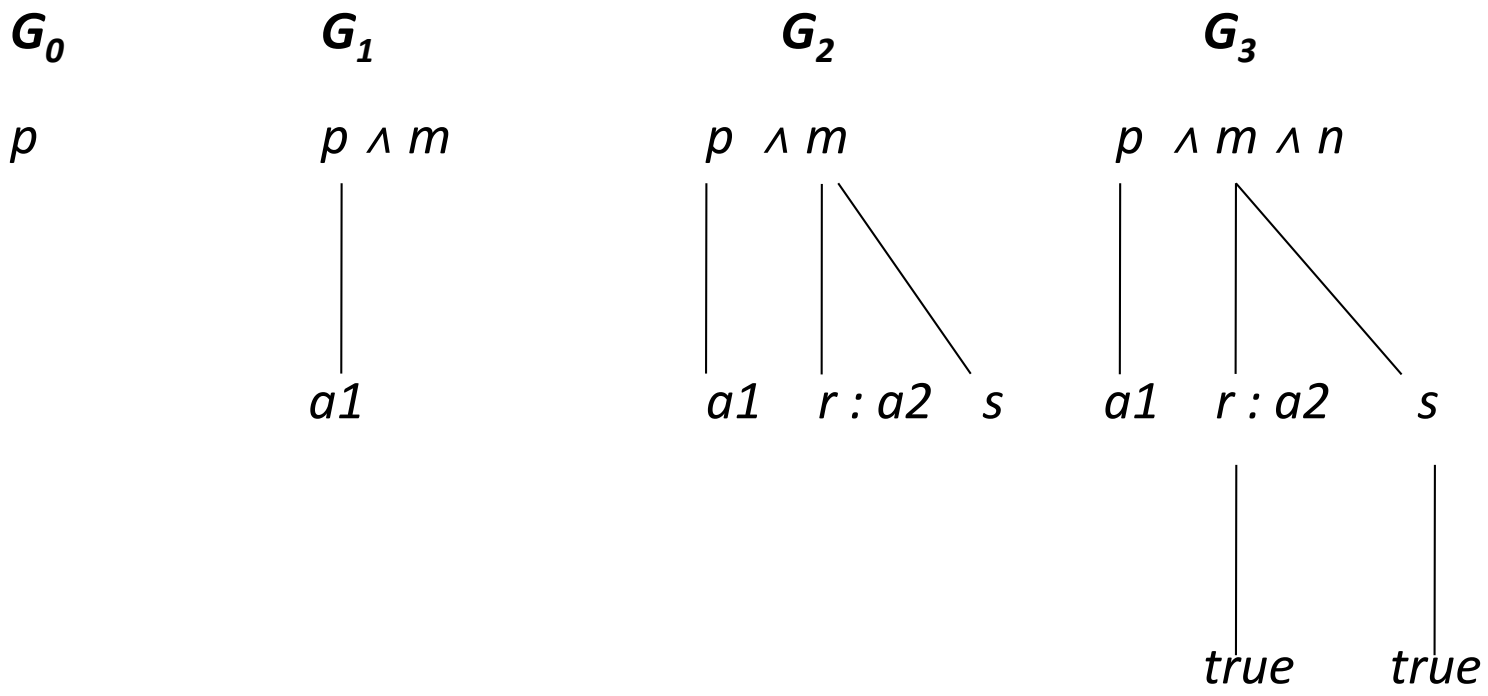
Because $Act = \{\}$, $E_0 = E_1 = E_2 = E_3$.

Cycle 1: Rule1 fires and adds m to the goal state,
and a solution for p is found using Defn 1
$G_1 = (p \vee a1) \wedge m.$

Cycle 2: Two alternative solutions for m are found using Defn 2 and Defn 3.
$a1$ is selected for execution.
$G_2 = (p \vee a1) \wedge (m \vee (r : a2) \vee s).$

Cycle 3: $a1$ is successfully executed, and resolved with $a1$ in the goal state,
s in the goal state is resolved with the database,
and a new top level goal n is generated by Rule 2.
$G_3 = (p \vee a1 \vee true) \wedge (m \vee (r : a2) \vee s \vee true) \wedge n$
$= n$

Goal states have the structure of an SLD resolution-like search tree



A selection function is *timely* if the selected subgoal is
a macro-action subgoal anywhere in a goal clause,
or a query subgoal or atomic action at the beginning of a goal clause.
An instance $a\sigma$ of an action a is *executable* if the query *possible(a)*
has answer σ in the current state of the database (given by $\boldsymbol{E_i} \cup \boldsymbol{I} \cup \boldsymbol{Act}$)

Extended resolution

Given an (extended) goal clause *C* with selected subgoal:

If the selected subgoal is a macro-action *m* and there is a definition *m' ← conditions, such that m* and *m'* have a most general unifier (mgu) θ, then the *resolvent* is $C\theta$ with $m\theta$ replaced by *conditions*θ.
The old subgoals in the resolvent inherit their order from the parent clauses, and the new subgoals *conditions*θ have the same relative order to the old subgoals as *m* has to the other subgoals in *C.*

Example: The resolvent of *q(X) : m(X, Y) ; n(Y)* and *m(c, Z) ← a1(c) ; a2(Z),*
is *q(c) : a1(c) ; a2(Y) ; n(Y).*

If the selected subgoal is a query or atomic action then it must be the first subgoal G_1 in *C*. If G_1 succeeds with mgu θ, then the *resolvent* is:

$: G_2\theta$	if *C* is $G_1 : G_2$	or *C* is $: G_1 : G_2$
$G_2\theta$	if *C* is $G_1 ; G_2$	or *C* is $: G_1 ; G_2$
true	if *C* is G_1	or *C* is $: G_1$

The Operational Semantics

Given an LPS framework $\langle \boldsymbol{I, R, Mac, Act} \rangle$ with a timely selection function, an initial state $\langle \boldsymbol{E_0, G_0} \rangle$ and an input stream of externally generated observations $O_0, , \ldots, O_i, \ O_{i+1}$, the agent cycle generates a sequence:

$$\langle \boldsymbol{E_0, G_0} \rangle, A_0, O_0, \langle \boldsymbol{E_1, G_1} \rangle , \ldots, \langle \boldsymbol{E_i, G_i} \rangle, A_i, O_i, \ldots.$$

Let $\mathbf{Max}_i$ be a bound on the number of resolutions performed in each cycle.

It suffices to specify the *transition* from one state $\langle \boldsymbol{E_i, G_i} \rangle, A_i, O_i$ to the next $\langle \boldsymbol{E_{i+1}, G_{i+1}} \rangle, \ A_{i+1}$:

The Operational Semantics

LPS0. Update the database: Let $Events_i = O_i \cup A_i$.
$\boldsymbol{E_{i+1}} = \boldsymbol{E_i}$ - $\{p : terminates(e, p)$ **holds in** $\boldsymbol{E_i} \cup \boldsymbol{I} \cup \boldsymbol{Act}$ and $e \in Events_i\} \cup$
$\{p : initiates(e, p)$ **holds in** $\boldsymbol{E_i} \cup \boldsymbol{I} \cup \boldsymbol{Act}$ and $e \in Events_i\}$.

LPS1. Generate new goals: For every *conditions* $\rightarrow$ *conclusion* in ***R***
such that *conditions* σ holds in $\boldsymbol{E_{i+1}} \cup \boldsymbol{I} \cup Events_i$,
a new goal tree is added to $\boldsymbol{G_i}$ with root *conclusion* σ.
Let ***GR*** be the resulting intermediate goal state.

LPS3. Plan and execute: Let ***GRR*** be the goal state starting from ***GR***
derivable by resolving with $\boldsymbol{Mac} \cup \boldsymbol{E_{i+1}} \cup \boldsymbol{I} \cup A_i$ within the bound $\mathbf{Max_i}$.

LPS3.1 If ***GRR*** does not contain a goal clause with a selected, executable action, then A_{i+1} is an empty set of actions.

LPS3.2 Otherwise, let A be a selected set of executable actions chosen by the search strategy. Then $A_{i+1} = \{ a\sigma$: $a \in A$ and $a\sigma$ *is executable}.*

LPS4. Prune the goal trees: All goal clauses in ***GRR*** prefixed by ***:*** not selected in this cycle are deleted, resulting in the goal state $\boldsymbol{G_{i+1}}$.

Computational Logic and Human Thinking – ALP agents and BDI agents compared

- Three kinds of rules
- Robot room-cleaning example – BDI vs ALP/LPS
- Teleo-reactivity
- LPS syntax and model-theoretic semantics
- LPS operational semantics
- Soundness and Incompleteness

Incompleteness – two sources

No preventative maintenance:

***R**: q → a1 .* ***Act**: terminates(a2, q).* $\boldsymbol{E_0}$ *: {q}*

where *a1* and *a2* are atomic actions. There is a minimal model corresponding to the sequence of actions *a2, a1*, but the LPS cycle can only generate the non-terminating sequence *a1, a1.*

Inadequate macro-action program:

***R**: q → p .* ***Act**: initiates(a, p).* $\boldsymbol{E_0}$ *: {q}*

There is a minimal model containing action *a*, but the LPS cycle cannot generate it.

Computational Logic and Human Thinking – ALP agents and BDI agents compared - Conclusions

- Production systems and BDI agents have a state-free syntax, destructive database, and no declarative semantics
- ALP agents have a syntax with time or state, a model-theoretic semantics, and a database without destructive updates
- LPS has a model-theoretic semantics, a state-free syntax, and a destructive database
- LPS is related to Nilsson's Teleo-reactive production systems, and Bonner and Kifer's TR logic.
- LPS is sound, but incomplete

Computational Logic and Human Thinking – Conclusions

The ALP agent model

- provides a descriptive and normative model of human thinking and deciding
- explains psychological experiments that claim to show that people are illogical
- includes a model of The Language of Thought (LOT)
- is compatible with classical logic but also has a minimal model semantics
- is compatible with the use of a destructive database and a state-free syntax

Computational Logic and Human Thinking - Conclusions

AI tools and techniques

- can be reconciled and be combined
- improve those of existing academic disciplines
- can be used by ordinary people