Logic, Inference and Computation

Clint van Alten

School of Computer Science
University of the Witwatersrand, Johannesburg
South Africa

June 24, 2014

Logic and Inference

$$\left.\begin{array}{l} \textit{Premise 1} \\ \textit{Premise 2} \\ \textit{Premise 3} \\ \quad\vdots \\ \textit{Premise } n \end{array}\right\} \Longrightarrow \textit{Conclusion (?)}$$

The *Premises* and the *Conclusion* are built up from some fixed *Alphabet*.

There is a fixed set of *Inference Rules* that we may apply in steps to derive the *Conclusion* from the *Premises*.

Logic and Inference

$$\left.\begin{array}{l}(\forall x, y, z)[(x \leq y \;\&\; y \leq z) \rightarrow x \leq z] \\ (\forall x, y)[(x \leq y \;\&\; y \leq x) \rightarrow x = y] \\ (\forall x)[x \leq x] \\ (\exists x)(\forall y)[x \leq y \text{ or } y \leq x]\end{array}\right\} \Longrightarrow (\exists w)(\forall x)[x \leq w]$$

Using Inference Rules of First-Order Logic.

Logic and Inference

$$\left.\begin{array}{l} \textit{Today is rainy.} \\ \textit{Kids are at school.} \\ \textit{No sport on TV.} \end{array}\right\} \Longrightarrow \textit{Go to the movies (?)}$$

$$\left.\begin{array}{l} \text{All men with grey beards are wise or old.} \\ \text{Some Profs are wise.} \\ \text{Some old men are wise.} \\ \textit{John} \text{ has a grey beard and is not old.} \end{array}\right\} \Longrightarrow \textit{John is a Prof (?)}$$

$$\text{Database } \} \Longrightarrow \text{Query (?)}$$

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: abc

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: abc

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc$

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc$

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc = bd$

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc = bd = ?$

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc = bd = ?$

Attempt 2: abc

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc = bd = ?$

Attempt 2: abc

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc = bd = ?$

Attempt 2: $abc = aabc$

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc = bd = ?$

Attempt 2: $abc = aabc$

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc = bd = ?$

Attempt 2: $abc = aabc = abcc$

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc = bd = ?$

Attempt 2: $abc = aabc = abcc$

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc = bd = ?$

Attempt 2: $abc = aabc = abcc = abd$

An example from Algebra - Word Problem for Semigroups

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: associativity: $(xy)z = x(yz)$, and substitution anywhere in a word.

Attempt 1: $abc = bcc = bd = ?$

Attempt 2: $abc = aabc = abcc = abd$!

An Algorithm

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \\ bda = ca \\ aa = cb \end{array}\right\} \Longrightarrow abcde = dbac\ (?)$$

An Algorithm

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \\ bda = ca \\ aa = cb \end{array}\right\} \Longrightarrow abcde = dbac\ (?)$$

$$abcde$$

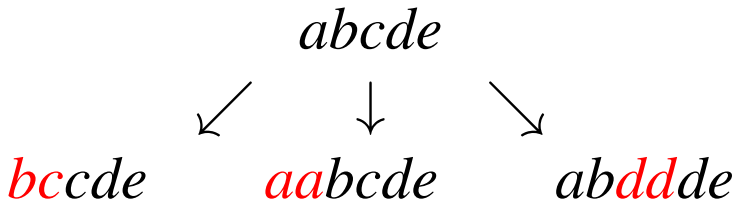
An Algorithm

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \\ bda = ca \\ aa = cb \end{array}\right\} \Longrightarrow abcde = dbac\ (?)$$

$$abcde$$
$$\swarrow \quad \downarrow \quad \searrow$$

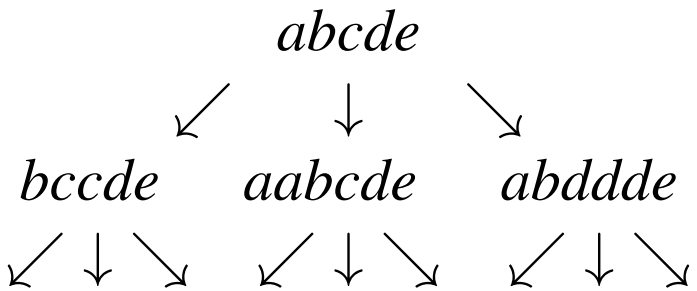
An Algorithm

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \\ bda = ca \\ aa = cb \end{array}\right\} \Longrightarrow abcde = dbac\ (?)$$



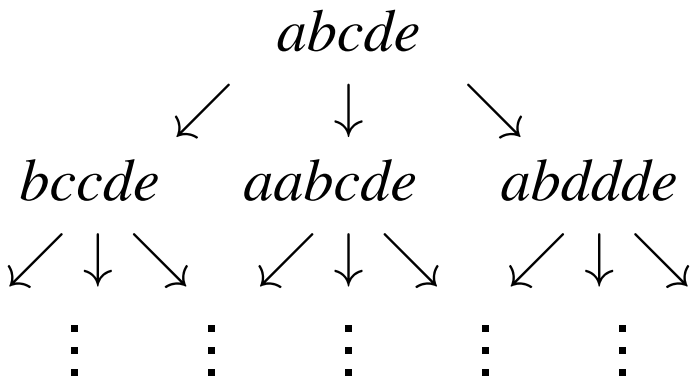
An Algorithm

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \\ bda = ca \\ aa = cb \end{array}\right\} \Longrightarrow abcde = dbac\ (?)$$



An Algorithm

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \\ bda = ca \\ aa = cb \end{array}\right\} \Longrightarrow abcde = dbac\ (?)$$



Word Problem for Semigroups

- If the *Conclusion* is derivable from the *Premises* then the algorithm will find a derivation.

- If the *Conclusion* is not derivable from the *Premises* then the algorithm will search forever, without halting.

Theorem

The Word Problem for Semigroups is undecidable.

Proof.

If it was decidable, then ..... ! # & ↩ρ □

Word Problem for Semigroups

- If the *Conclusion* is derivable from the *Premises* then the algorithm will find a derivation.

- If the *Conclusion* is not derivable from the *Premises* then the algorithm will search forever, without halting.

Theorem

The Word Problem for Semigroups is undecidable.

Proof.

If it was decidable, then ..... ! # & ↩ρ □

Word Problem for Semigroups

- If the *Conclusion* is derivable from the *Premises* then the algorithm will find a derivation.

- If the *Conclusion* is not derivable from the *Premises* then the algorithm will search forever, without halting.

Theorem

The Word Problem for Semigroups is undecidable.

Proof.

If it was decidable, then ..... ! # & ↩ρ □

Word Problem for Semigroups

- If the *Conclusion* is derivable from the *Premises* then the algorithm will find a derivation.

- If the *Conclusion* is not derivable from the *Premises* then the algorithm will search forever, without halting.

Theorem

The Word Problem for Semigroups is undecidable.

Proof.

If it was decidable, then ..... ! # & ↩℘ □

How to decide non-implication

How can we decide if:

$$\left.\begin{array}{c} \textit{Premise 1} \\ \textit{Premise 2} \\ \vdots \\ \textit{Premise } n \end{array}\right\} \not\Longrightarrow \textit{Conclusion}$$

- We try to build a model or a world in which all *Premises* are true but the *Conclusion* is false.
- If such a model exists, then the *Conclusion* does not follow from the *Premises*.
- If the *Conclusion* does follow from the *Premises*, then in every model in which all *Premises* are true, the *Conclusion* must also be true.

How to decide non-implication

How can we decide if:

$$\left.\begin{array}{c} \textit{Premise 1} \\ \textit{Premise 2} \\ \vdots \\ \textit{Premise } n \end{array}\right\} \not\Longrightarrow \textit{Conclusion}$$

- We try to build a model or a world in which all *Premises* are true but the *Conclusion* is false.
- If such a model exists, then the *Conclusion* does not follow from the *Premises*.
- If the *Conclusion* does follow from the *Premises*, then in every model in which all *Premises* are true, the *Conclusion* must also be true.

How to decide non-implication

How can we decide if:

$$\left.\begin{array}{c} \textit{Premise 1} \\ \textit{Premise 2} \\ \vdots \\ \textit{Premise } n \end{array}\right\} \not\Longrightarrow \textit{Conclusion}$$

- We try to build a model or a world in which all *Premises* are true but the *Conclusion* is false.
- If such a model exists, then the *Conclusion* does not follow from the *Premises*.
- If the *Conclusion* does follow from the *Premises*, then in every model in which all *Premises* are true, the *Conclusion* must also be true.

How to decide non-implication

How can we decide if:

$$\left.\begin{array}{c} \textit{Premise 1} \\ \textit{Premise 2} \\ \vdots \\ \textit{Premise } n \end{array}\right\} \not\Longrightarrow \textit{Conclusion}$$

- We try to build a model or a world in which all *Premises* are true but the *Conclusion* is false.
- If such a model exists, then the *Conclusion* does not follow from the *Premises*.
- If the *Conclusion* does follow from the *Premises*, then in every model in which all *Premises* are true, the *Conclusion* must also be true.

A algorithm for deciding non-inference

Start building all possible models and, for each one, check the *Premises* and the *Conclusion*:

If we find a model in which

- all *Premises* are true and
- the *Conclusion* is false,

then the inference fails. This is called a countermodel.

We want the computer to do the model building and model-checking, so the models must be computable.

Finite Model Property:
If the inference fails, then it fails in a *finite* world.

A algorithm for deciding non-inference

Start building all possible models and, for each one, check the *Premises* and the *Conclusion*:

If we find a model in which

- all *Premises* are true and
- the *Conclusion* is false,

then the inference fails. This is called a countermodel.

We want the computer to do the model building and model-checking, so the models must be computable.

Finite Model Property:
If the inference fails, then it fails in a *finite* world.

A algorithm for deciding non-inference

Start building all possible models and, for each one, check the *Premises* and the *Conclusion*:

If we find a model in which

- all *Premises* are true and
- the *Conclusion* is false,

then the inference fails. This is called a countermodel.

We want the computer to do the model building and model-checking, so the models must be computable.

Finite Model Property:
If the inference fails, then it fails in a *finite* world.

Parallel algorithms for Decidability

Given the input:

$$\textit{Premise}\ 1, \ldots, \textit{Premise}\ n, \quad \textit{Conclusion}$$

we run two algorithms in parallel:

Search for a derivation of *Conclusion* from *Premises*	Search for a finite countermodel to *Premises* and *Conclusion*
$\Downarrow$	$\Downarrow$

One of these algorithms is guaranteed to halt.
'don't call me, I'll call you...'

Parallel algorithms for Decidability

Given the input:

$$\textit{Premise}\ 1, \ldots, \textit{Premise}\ n, \quad \textit{Conclusion}$$

we run two algorithms in parallel:

Search for a derivation of *Conclusion* from *Premises*	Search for a finite countermodel to *Premises* and *Conclusion*
$\Downarrow$	$\Downarrow$

One of these algorithms is guaranteed to halt.
'don't call me, I'll call you...'

The next step - Complexity

Word Problem for Commutative Semigroups.

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: we also have commutativity $xy = yx$.

This problem is decidable,

- but what is its computational complexity?

If the size of the input is ℓ:

$$\underbrace{\textit{Premise}\ 1, \ldots, \textit{Premise}\ n, \quad \textit{Conclusion}}_{\ell}$$

the maximum number of steps for a derivation is about $2^{2^{\ell}}$.

The next step - Complexity

Word Problem for Commutative Semigroups.

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: we also have commutativity $xy = yx$.

This problem is decidable,

- but what is its computational complexity?

If the size of the input is ℓ:

$$\underbrace{\textit{Premise}\ 1, \ldots, \textit{Premise}\ n, \quad \textit{Conclusion}}_{\ell}$$

the maximum number of steps for a derivation is about $2^{2^{\ell}}$.

The next step - Complexity

Word Problem for Commutative Semigroups.

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: we also have commutativity $xy = yx$.

This problem is decidable,

- but what is its computational complexity?

If the size of the input is ℓ:

$$\underbrace{\textit{Premise } 1, \ldots, \textit{Premise } n, \quad \textit{Conclusion}}_{\ell}$$

the maximum number of steps for a derivation is about $2^{2^{\ell}}$.

The next step - Complexity

Word Problem for Commutative Semigroups.

$$\left.\begin{array}{l} ab = bc \\ cc = d \\ aa = a \end{array}\right\} \Longrightarrow abc = abd \quad (?)$$

Inference Rules: we also have commutativity $xy = yx$.

This problem is decidable,

- but what is its computational complexity?

If the size of the input is ℓ:

$$\underbrace{\textit{Premise}\ 1, \ldots, \textit{Premise}\ n, \quad \textit{Conclusion}}_{\ell}$$

the maximum number of steps for a derivation is about $2^{2^{\ell}}$.

My research interests

- the study of decidability and complexity for problems related to algebra and logic
- the use of partial algebras in proving finite model properties
- this method has been applied to obtain decidability or complexity of the word problem for distributive lattices, Heyting algebras and various classes of residuated lattices.
- related problems such as finding other model constructions, finding good systems of inference rules, algebraic properties...

Real-World Applications

All men with grey beards are wise or old.
Some Profs are wise.
Some old men are wise.
John has a grey beard and is not old.

$\Longrightarrow$ *John is a Prof (?)*

We may think of the *Premises* here as a knowledge base and the *Conclusion* as a query.

We can write the above using First-Order Logic as:

$$\left.\begin{array}{l}(\forall x)[GB(x) \rightarrow (W(x) \text{ or } O(x))] \\ (\exists x)[P(x)\ \&\ W(x)] \\ (\exists x)[O(x)\ \&\ W(x)] \\ GB(John)\ \&\ \text{not}\, O(John)\end{array}\right\} \Longrightarrow P(John)\ (?)$$

Real-World Applications

All men with grey beards are wise or old.
Some Profs are wise.
Some old men are wise.
John has a grey beard and is not old.

$\Longrightarrow$ *John is a Prof (?)*

We may think of the *Premises* here as a knowledge base and the *Conclusion* as a query.

We can write the above using First-Order Logic as:

$$\left.\begin{array}{l}(\forall x)[GB(x) \rightarrow (W(x) \text{ or } O(x))] \\ (\exists x)[P(x)\ \&\ W(x)] \\ (\exists x)[O(x)\ \&\ W(x)] \\ GB(John)\ \&\ \text{not}\, O(John)\end{array}\right\} \Longrightarrow P(John)\ (?)$$

Automated Reasoning

First-Order Logic is a very expressive language, but...

- First-Order Logic is undecidable.

By restricting the type of sentences allowed in the knowledge base and in the queries, we may get decidable systems.

- There is always a trade-off between expressivity and complexity/decidability.

One can custom-make logical systems that are suited for very specific purposes, e.g., software or hardware verification, planning, database queries, etc. (Esp. Modal Logic)

- If we remove the quantifiers we have propositional logic, which is a decidable, NP-complete logic.
- If we then allow only Horn clauses, we get a logic with linear-time complexity.

Automated Reasoning

First-Order Logic is a very expressive language, but...

- First-Order Logic is undecidable.

By restricting the type of sentences allowed in the knowledge base and in the queries, we may get decidable systems.

- There is always a trade-off between expressivity and complexity/decidability.

One can custom-make logical systems that are suited for very specific purposes, e.g., software or hardware verification, planning, database queries, etc. (Esp. Modal Logic)

- If we remove the quantifiers we have propositional logic, which is a decidable, NP-complete logic.
- If we then allow only Horn clauses, we get a logic with linear-time complexity.

Automated Reasoning

First-Order Logic is a very expressive language, but...

- First-Order Logic is undecidable.

By restricting the type of sentences allowed in the knowledge base and in the queries, we may get decidable systems.

- There is always a trade-off between expressivity and complexity/decidability.

One can custom-make logical systems that are suited for very specific purposes, e.g., software or hardware verification, planning, database queries, etc. (Esp. Modal Logic)

- If we remove the quantifiers we have propositional logic, which is a decidable, NP-complete logic.
- If we then allow only Horn clauses, we get a logic with linear-time complexity.

Automated Reasoning

First-Order Logic is a very expressive language, but...

- First-Order Logic is undecidable.

By restricting the type of sentences allowed in the knowledge base and in the queries, we may get decidable systems.

- There is always a trade-off between expressivity and complexity/decidability.

One can custom-make logical systems that are suited for very specific purposes, e.g., software or hardware verification, planning, database queries, etc. (Esp. Modal Logic)

- If we remove the quantifiers we have propositional logic, which is a decidable, NP-complete logic.
- If we then allow only Horn clauses, we get a logic with linear-time complexity.

Automated Reasoning

First-Order Logic is a very expressive language, but...

- First-Order Logic is undecidable.

By restricting the type of sentences allowed in the knowledge base and in the queries, we may get decidable systems.

- There is always a trade-off between expressivity and complexity/decidability.

One can custom-make logical systems that are suited for very specific purposes, e.g., software or hardware verification, planning, database queries, etc. (Esp. Modal Logic)

- If we remove the quantifiers we have propositional logic, which is a decidable, NP-complete logic.
- If we then allow only Horn clauses, we get a logic with linear-time complexity.

Automated Reasoning

First-Order Logic is a very expressive language, but...

- First-Order Logic is undecidable.

By restricting the type of sentences allowed in the knowledge base and in the queries, we may get decidable systems.

- There is always a trade-off between expressivity and complexity/decidability.

One can custom-make logical systems that are suited for very specific purposes, e.g., software or hardware verification, planning, database queries, etc. (Esp. Modal Logic)

- If we remove the quantifiers we have propositional logic, which is a decidable, NP-complete logic.
- If we then allow only Horn clauses, we get a logic with linear-time complexity.

Automated Theorem Provers - Robbins Algebras

The following problem was posed in 1933:

$$\left.\begin{array}{l}(\forall x, y, z)[x \vee (y \vee z) = (x \vee y) \vee z] \\ (\forall x, y)[x \vee y = y \vee x] \\ (\forall x, y)[\neg(\neg(x \vee y) \vee \neg(x \vee \neg y)) = x]\end{array}\right\} \Longrightarrow$$

$$(\forall x, y)[\neg(\neg x \vee y) \vee \neg(\neg x \vee \neg y) = x]$$

Using Inference Rules of First-Order Logic.

A proof of the above inference was found 1996 by William McCune, using an automated theorem prover

- the derivation required 8900 steps.

Automated Theorem Provers - Robbins Algebras

The following problem was posed in 1933:

$$\left.\begin{array}{l}(\forall x, y, z)[x \vee (y \vee z) = (x \vee y) \vee z] \\ (\forall x, y)[x \vee y = y \vee x] \\ (\forall x, y)[\neg(\neg(x \vee y) \vee \neg(x \vee \neg y)) = x]\end{array}\right\} \Longrightarrow$$

$$(\forall x, y)[\neg(\neg x \vee y) \vee \neg(\neg x \vee \neg y) = x]$$

Using Inference Rules of First-Order Logic.

A proof of the above inference was found 1996 by William McCune, using an automated theorem prover

- the derivation required 8900 steps.

Automated Theorem Provers - Robbins Algebras

The following problem was posed in 1933:

$$\left.\begin{array}{l}(\forall x, y, z)[x \vee (y \vee z) = (x \vee y) \vee z] \\ (\forall x, y)[x \vee y = y \vee x] \\ (\forall x, y)[\neg(\neg(x \vee y) \vee \neg(x \vee \neg y)) = x]\end{array}\right\} \Longrightarrow$$

$$(\forall x, y)[\neg(\neg x \vee y) \vee \neg(\neg x \vee \neg y) = x]$$

Using Inference Rules of First-Order Logic.

A proof of the above inference was found 1996 by William McCune, using an automated theorem prover

- the derivation required 8900 steps.

Uncertain Information

How to deal with statements that are neither true nor false:

- 'Today is a cloudy day.'
- Rover is a big dog.'
- 'Ebrahim is a good Head of School.'
- 'Holland will win the Soccer World Cup.'

- 'Today is a cloudy day.' (0.4)
- Rover is a big dog.' (0.7)
- 'Ebrahim is a good Head of School.' (?)
- 'Holland will win the Soccer World Cup.' 0.9)

Uncertain Information

How to deal with statements that are neither true nor false:

- 'Today is a cloudy day.'
- Rover is a big dog.'
- 'Ebrahim is a good Head of School.'
- 'Holland will win the Soccer World Cup.'

- 'Today is a cloudy day.' (0.4)
- Rover is a big dog.' (0.7)
- 'Ebrahim is a good Head of School.' (?)
- 'Holland will win the Soccer World Cup.' 0.9)

Reasoning with Degrees of Belief

$$\left.\begin{array}{ll} \textit{Premise 1} & \textit{(0.9)} \\ \textit{Premise 2} & \textit{(0.75)} \\ \textit{Premise 3} & \textit{(0.1)} \\ \vdots & \\ \textit{Premise } n & \textit{(0.8)} \end{array}\right\} \Longrightarrow \textit{Conclusion (0.9)}$$

The number associated with each statement can be interpretted as:

- a Degree of Belief in the statement, or
- the Probability that the statement is true.

Either way requires a new kind a logic inference rules:

- Many-Valued Logics
- Inductive Logic
- Bayesian Logic

Reasoning with Degrees of Belief

$$\left.\begin{array}{ll} \textit{Premise 1} & \textit{(0.9)} \\ \textit{Premise 2} & \textit{(0.75)} \\ \textit{Premise 3} & \textit{(0.1)} \\ \vdots & \\ \textit{Premise } n & \textit{(0.8)} \end{array}\right\} \Longrightarrow \textit{Conclusion (0.9)}$$

The number associated with each statement can be interpretted as:

- a Degree of Belief in the statement, or
- the Probability that the statement is true.

Either way requires a new kind a logic inference rules:

- Many-Valued Logics
- Inductive Logic
- Bayesian Logic

Reasoning with Degrees of Belief

$$\left.\begin{array}{ll} \textit{Premise 1} & \textit{(0.9)} \\ \textit{Premise 2} & \textit{(0.75)} \\ \textit{Premise 3} & \textit{(0.1)} \\ \vdots & \\ \textit{Premise } n & \textit{(0.8)} \end{array}\right\} \Longrightarrow \textit{Conclusion (0.9)}$$

The number associated with each statement can be interpretted as:

- a Degree of Belief in the statement, or
- the Probability that the statement is true.

Either way requires a new kind a logic inference rules:

- Many-Valued Logics
- Inductive Logic
- Bayesian Logic

Inference on Datasets

Example: Pricing property:

$$\left.\begin{array}{l} \textit{size of property} = 900\ m^2 \\ \textit{numbers of bedrooms} = 3 \\ \textit{condition} = 85\% \\ \textit{quality of area} = 70\% \\ \textit{distance to city centre} = 2km \end{array}\right\} \Longrightarrow \text{Price} = \text{R1, 200,000}$$

$$\left.\begin{array}{l} \textit{size of property} = 920\ m^2 \\ \textit{numbers of bedrooms} = 3 \\ \textit{condition} = 90\% \\ \textit{quality of area} = 65\% \\ \textit{distance to city centre} = 3km \end{array}\right\} \Longrightarrow \text{Price} = \text{R??}$$

Inference on Datasets

Example: Pricing property:

$$\left.\begin{array}{l} \textit{size of property} = 900\ m^2 \\ \textit{numbers of bedrooms} = 3 \\ \textit{condition} = 85\% \\ \textit{quality of area} = 70\% \\ \textit{distance to city centre} = 2km \end{array}\right\} \Longrightarrow \text{Price} = \text{R1, 200,000}$$

$$\left.\begin{array}{l} \textit{size of property} = 920\ m^2 \\ \textit{numbers of bedrooms} = 3 \\ \textit{condition} = 90\% \\ \textit{quality of area} = 65\% \\ \textit{distance to city centre} = 3km \end{array}\right\} \Longrightarrow \text{Price} = \text{R??}$$

Inference on Datasets

$$\left.\begin{array}{cc} State_1, & Outcome_1 \\ \vdots & \vdots \\ State_n, & Outcome_n \end{array}\right\} \Longrightarrow \text{for a given State predict } Outcome$$

$$\left.\begin{array}{ccc} State_1, & Action_1, & Outcome_1 \\ \vdots & \vdots & \vdots \\ State_n, & Action_n, & Outcome_n \end{array}\right\} \Longrightarrow$$

- for given *State* and *Action* predict *Outcome*
- for given *State* find *Action* that optimizes *Outcome*

Inference on Datasets

$$\left.\begin{array}{cc} State_1, & Outcome_1 \\ \vdots & \vdots \\ State_n, & Outcome_n \end{array}\right\} \Longrightarrow \text{for a given State predict } Outcome$$

$$\left.\begin{array}{ccc} State_1, & Action_1, & Outcome_1 \\ \vdots & \vdots & \vdots \\ State_n, & Action_n, & Outcome_n \end{array}\right\} \Longrightarrow$$

- for given *State* and *Action* predict *Outcome*
- for given *State* find *Action* that optimizes *Outcome*

Machine Learning Models

Build a mathematical model for the data:

- A function f that takes as input *State* and returns an estimate for the *Outcome*.
- The function f should have some built-in unknown parameters w_1, w_2, w_3, w_4 that can vary.
- Different parameters correspond to different models.

Using an algorithm, we try to find a model that fits the data well:

- through incremental changes to the parameter values, or
- by solving an optimization problem to get the best parameter values.

Then we use the final model to predict *Outcomes* for any inputs.

Machine Learning Models

Build a mathematical model for the data:

- A function f that takes as input *State* and returns an estimate for the *Outcome*.
- The function f should have some built-in unknown parameters w_1, w_2, w_3, w_4 that can vary.
- Different parameters correspond to different models.

Using an algorithm, we try to find a model that fits the data well:

- through incremental changes to the parameter values, or
- by solving an optimization problem to get the best parameter values.

Then we use the final model to predict *Outcomes* for any inputs.

Recent MSc projects

- Learning medical treatment strategies based on recorded treatments and outcomes. (Sonali)
- Predicting missing information in land-surface data obtained from the MISR satellite. (Adrian)
- Pricing stock options based on a set of historical options or randomly generated stock trajectories. (Kevin)
- Learning to mimic expert behaviour in computer generated music. (Orry)
- Creating realistic simulations of multi-agent crowd behaviour. (Fred)
- Investigating and experimenting with new types of learning algorithms. (Pravesh and Dean) - PhD projects

Recent MSc projects

- Learning medical treatment strategies based on recorded treatments and outcomes. (Sonali)
- Predicting missing information in land-surface data obtained from the MISR satellite. (Adrian)
- Pricing stock options based on a set of historical options or randomly generated stock trajectories. (Kevin)
- Learning to mimic expert behaviour in computer generated music. (Orry)
- Creating realistic simulations of multi-agent crowd behaviour. (Fred)
- Investigating and experimenting with new types of learning algorithms. (Pravesh and Dean) - PhD projects

Recent MSc projects

- Learning medical treatment strategies based on recorded treatments and outcomes. (Sonali)
- Predicting missing information in land-surface data obtained from the MISR satellite. (Adrian)
- Pricing stock options based on a set of historical options or randomly generated stock trajectories. (Kevin)
- Learning to mimic expert behaviour in computer generated music. (Orry)
- Creating realistic simulations of multi-agent crowd behaviour. (Fred)
- Investigating and experimenting with new types of learning algorithms. (Pravesh and Dean) - PhD projects

Recent MSc projects

- Learning medical treatment strategies based on recorded treatments and outcomes. (Sonali)
- Predicting missing information in land-surface data obtained from the MISR satellite. (Adrian)
- Pricing stock options based on a set of historical options or randomly generated stock trajectories. (Kevin)
- Learning to mimic expert behaviour in computer generated music. (Orry)
- Creating realistic simulations of multi-agent crowd behaviour. (Fred)
- Investigating and experimenting with new types of learning algorithms. (Pravesh and Dean) - PhD projects

Recent MSc projects

- Learning medical treatment strategies based on recorded treatments and outcomes. (Sonali)
- Predicting missing information in land-surface data obtained from the MISR satellite. (Adrian)
- Pricing stock options based on a set of historical options or randomly generated stock trajectories. (Kevin)
- Learning to mimic expert behaviour in computer generated music. (Orry)
- Creating realistic simulations of multi-agent crowd behaviour. (Fred)
- Investigating and experimenting with new types of learning algorithms. (Pravesh and Dean) - PhD projects

Recent MSc projects

- Learning medical treatment strategies based on recorded treatments and outcomes. (Sonali)
- Predicting missing information in land-surface data obtained from the MISR satellite. (Adrian)
- Pricing stock options based on a set of historical options or randomly generated stock trajectories. (Kevin)
- Learning to mimic expert behaviour in computer generated music. (Orry)
- Creating realistic simulations of multi-agent crowd behaviour. (Fred)
- Investigating and experimenting with new types of learning algorithms. (Pravesh and Dean) - PhD projects

A word of thanks

Thanks to supervisors, collaborators, students, and management at wits university.

THE END

THE END - Thanks for attending !!