TAL 3.6 Discourse and Dialogue
Masters Course, Nancy University, 2007

Lecture 4: Fundamental inference tasks

Patrick Blackburn
TALARIS team
INRIA Nancy Grand Est
patrick.blackburn@loria.fr

17 October 2008

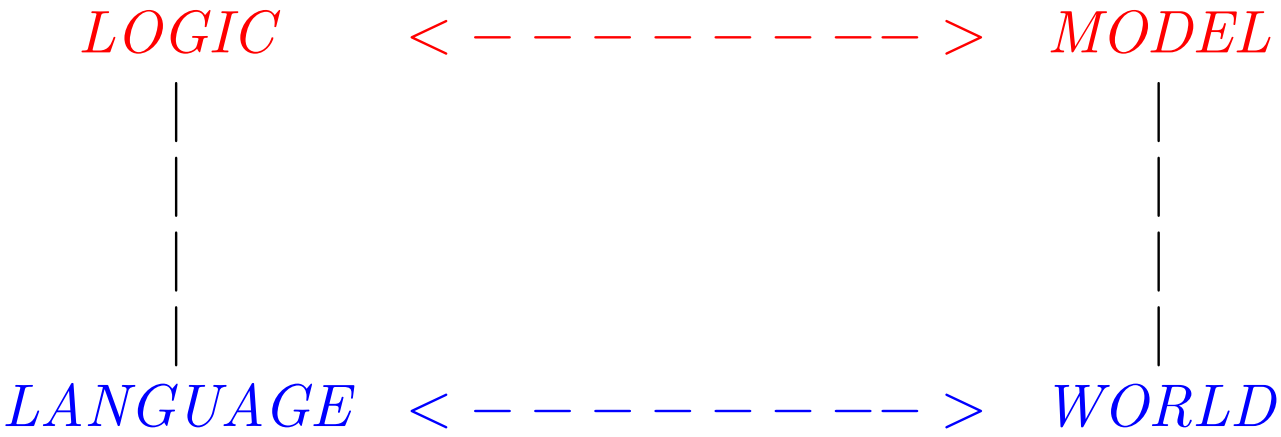
Where we are working

This course explores meaning in natural language from computational perspective.

$$LANGUAGE \; <-------> \; WORLD$$

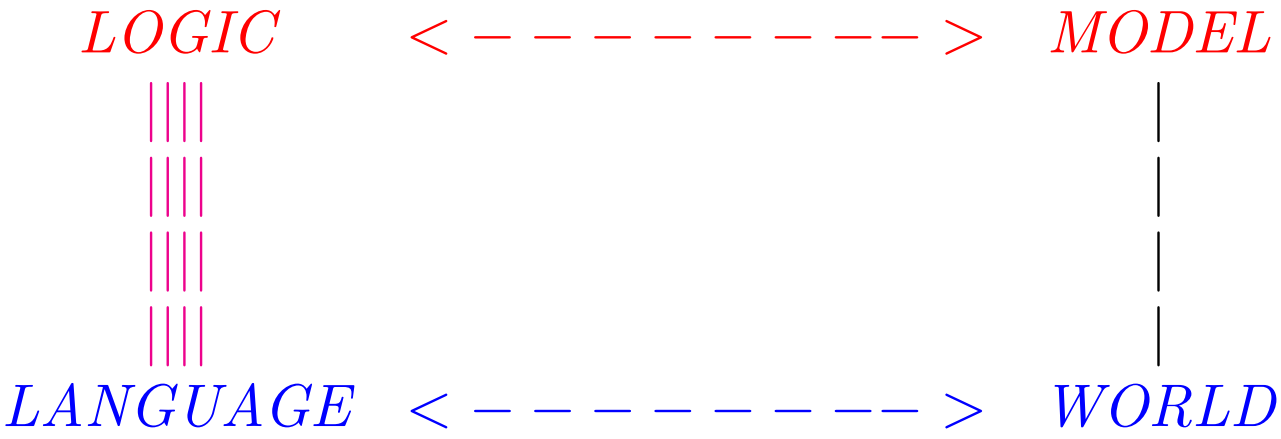
Where we are working

A basic them of the course is that logic provides a mathematical framework for such an exploration.



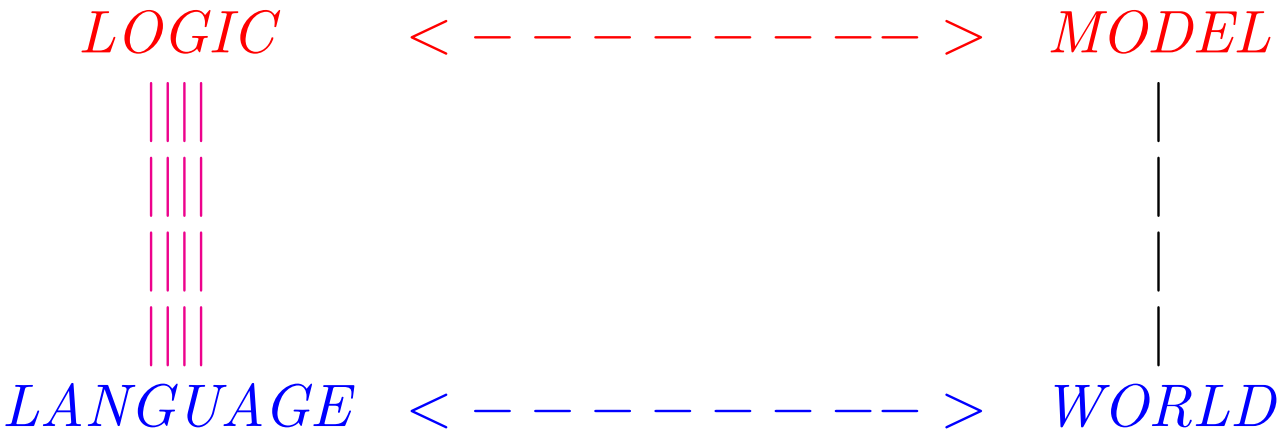
Where we are working

In the second lecture we explored the left-hand-side of the diagram:



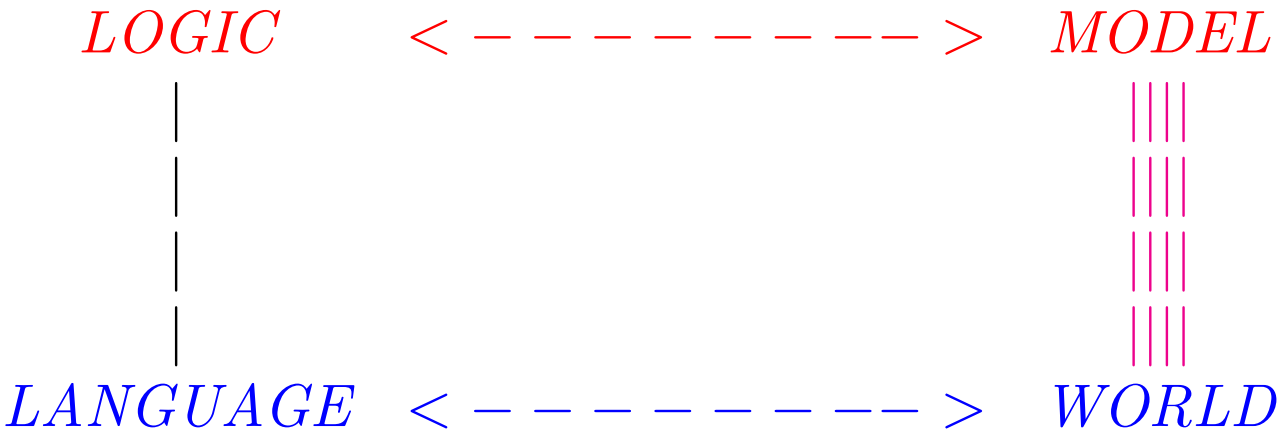
Where we are working

We saw that the link between logic and language can be made tight by translating language into logic.



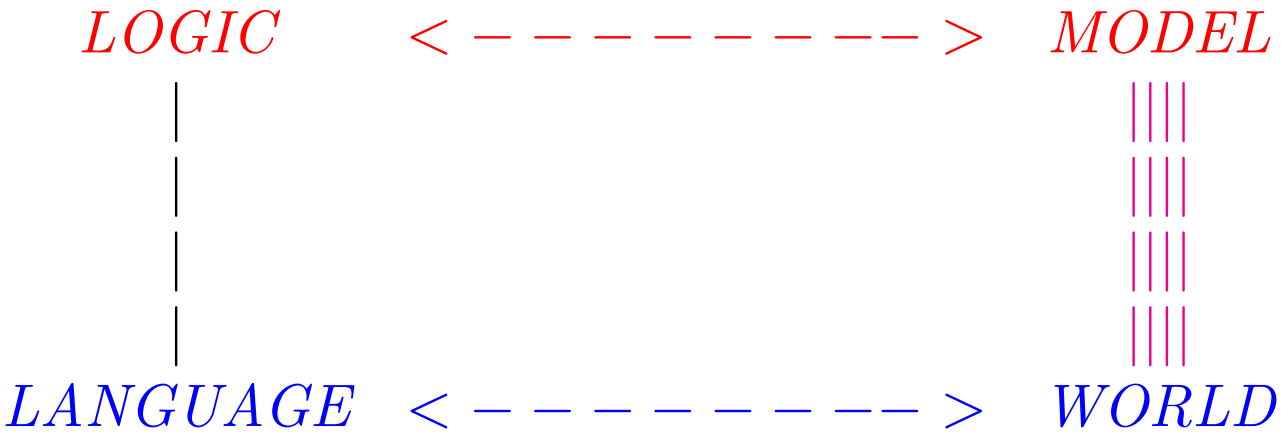
Where we are working

In the third lecture we explored the right-hand-side of the diagram:



Where we are working

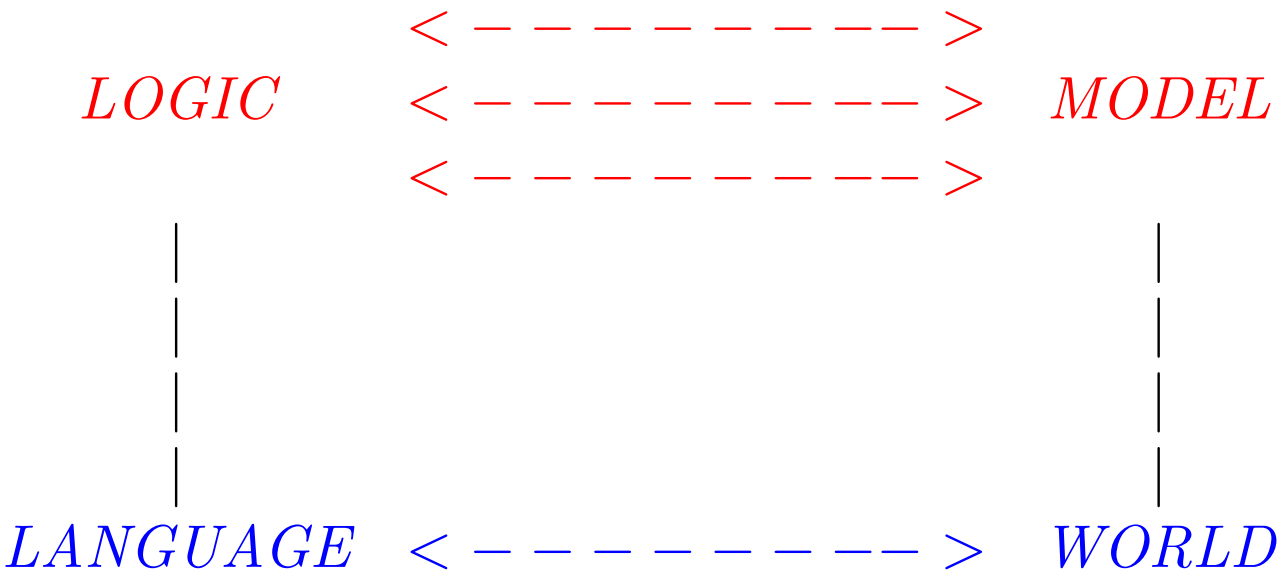
We talked about natural language metaphysics: defining models with the structure needed for coping with natural language.



So what are we doing today?

So what are we doing today?

We're looking at this part of the diagram:



Goals

- Get clearer about our two basic inference tasks — why they are important, and how we should think about them.
- Distinguish the proof theoretic approach to logic from the model-theoretic approach to logic.
- That is, distinguish the syntactic approach to logic from the semantic approach to logic.
- But the aim is not merely to *distinguish* them — that's the easy part. One of the deeper aims of today's lecture is to *show how they fit together*.
- A further aim of the lecture is to discuss these issues from a computational perspective — both theoretical and applied.

Key concepts

The key new logical concepts that will be introduced today are:

- Soundness
- Completeness

We will also discuss two more general concepts (that is, concepts that transcend logic) namely

- Decidability
- Undecidability

Two key technologies

We will also discuss two key inference technologies, namely

- Theorem proving
- Model building

and see what they have to offer us.

What about inference?

Up till now we've been talking a lot about representations and how to build them. But another other key word in computational semantics is inference. Why is this such an important word?

The answer is simple: inference seems ubiquitous when human being use natural language. We (seemingly effortlessly) make decisions about how to interpret and choose their use of language that require sophisticated use of inference.

Let's look at some simple examples which make use of two of the most fundamental inference tasks: consistency checking and informativity checking.

Consistency checking

Mia smokes.

Consistency checking

Mia smokes.
Mia does not smoke.

Consistency checking

Mia smokes.
Mia does not smoke.

Should be possible to detect the inconsistency in such discourses (and to avoid detecting inconsistency in superficially similar discourses such as Mia smokes. Mia did not smoke.)

Is Mia a tree?

Mia is a beautiful woman.

Is Mia a tree?

Mia is a beautiful woman.
Mia is a tree.

Is Mia a tree?

Mia is a beautiful woman.
Mia is a tree.

Consistency checking — but against background knowledge.

(The background knowledge used here might be such items as *Mia is a human being* and *Human beings are not plants*, and *Trees are plants*.)

Inference interacts with representation

Every boxer loves a woman.

$\forall x(\text{boxer}(x) \rightarrow \exists y(\text{woman}(y) \wedge \text{loves}(x, y)))$

$\exists y(\text{woman}(y) \wedge \forall x(\text{boxer}(x) \rightarrow \text{loves}(x, y)))$

She is extraordinarily beautiful.

They call her the Goddess of the Ring.

Resolving scope with consistency checks

Resolving scope with consistency checks

Every boxer has a broken nose.

Resolving scope with consistency checks

Every boxer has a broken nose.

$\forall x(\text{boxer}(x) \rightarrow \exists y(\text{broken-nose}(y) \wedge \text{has}(x, y)))$

Resolving scope with consistency checks

Every boxer has a broken nose.

$\forall x(\text{boxer}(x) \rightarrow \exists y(\text{broken-nose}(y) \wedge \text{has}(x, y)))$

??? $\exists y(\text{broken-nose}(y) \wedge \forall x(\text{boxer}(x) \rightarrow \text{has}(x, y)))$???

Biologically implausible — inconsistent with world knowledge

Resolving scope with consistency checks

Every boxer has a broken nose.

$\forall x(\text{boxer}(x) \rightarrow \exists y(\text{broken-nose}(y) \wedge \text{has}(x, y)))$

??? $\exists y(\text{broken-nose}(y) \wedge \forall x(\text{boxer}(x) \rightarrow \text{has}(x, y)))$???

Biologically implausible — inconsistent with world knowledge

Every car has a radio.

Resolving scope with consistency checks

Every boxer has a broken nose.

$\forall x(\text{boxer}(x) \rightarrow \exists y(\text{broken-nose}(y) \wedge \text{has}(x, y)))$

??? $\exists y(\text{broken-nose}(y) \wedge \forall x(\text{boxer}(x) \rightarrow \text{has}(x, y)))$???

Biologically implausible — inconsistent with world knowledge

Every car has a radio.

$\forall x(\text{car}(x) \rightarrow \exists y(\text{radio}(y) \wedge \text{has}(x, y)))$

Resolving scope with consistency checks

Every boxer has a broken nose.

$\forall x(\text{boxer}(x) \rightarrow \exists y(\text{broken-nose}(y) \wedge \text{has}(x, y)))$

??? $\exists y(\text{broken-nose}(y) \wedge \forall x(\text{boxer}(x) \rightarrow \text{has}(x, y)))$???

Biologically implausible — inconsistent with world knowledge

Every car has a radio.

$\forall x(\text{car}(x) \rightarrow \exists y(\text{radio}(y) \wedge \text{has}(x, y)))$

??? $\exists y(\text{radio}(y) \wedge \forall x(\text{car}(x) \rightarrow \text{has}(x, y)))$???

Possible in contexts with one car or no cars — otherwise inconsistent with world knowledge

Informativity checking

Informativity checking

Mia smokes.

Informativity checking

Mia smokes.
Mia smokes.

Informativity checking

Mia smokes.
Mia smokes.
Mia smokes.

Informativity checking

Mia smokes.
Mia smokes.
Mia smokes.

> *Make your contribution as informative as is required (for the current purposes of the exchange). H. P. Grice.*

Suggests need for Informativity checking — indeed, for informativity checking against background knowledge.

Informativity a ‘soft’ signal

Informativity a 'soft' signal

Mia is married.

Informativity a ‘soft’ signal

Mia is married.
She has a husband.

Informativity a 'soft' signal

Mia is married.
She has a husband.

Such a discourse could conform to Grice's requirement in many cases. The second utterance, though it merely rephrases the first, might well be "as informative as is required (for the current purposes of the exchange)".

So uninformativity a 'softer' signal than inconsistency (and it is not always clear what we should do once we've found it) — but it is something we would like to be able to detect.

Consistency and Informativity checks for Presupposition

A crucial part of Van der Sandt's Discourse Representation Theory, based presupposition resolution algorithm (which we shall discuss later in the course) is a structured sequence of consistency and informativity checks on subparts of the semantic representation.

Jean regrets that Marie is pregnant **PRESUPP** *Marie is pregnant*

Jean does not regret that Marie is pregnant **PRESUPP** *Marie is pregnant*

That is, consistency and informativity checks can be used as building blocks to analyse more sophisticated semantic tasks.

Consistency checking task in first-order logic

Consistency checking task in first-order logic

Let ϕ be the (first-order) semantic representation of the latest sentence in some ongoing discourse, and suppose that the relevant lexical knowledge, world knowledge, natural language metaphysical assumption, and the information from the previous discourse has been represented in first-order logic.

Consistency checking task in first-order logic

Let ϕ be the (first-order) semantic representation of the latest sentence in some ongoing discourse, and suppose that the relevant lexical knowledge, world knowledge, natural language metaphysical assumption, and the information from the previous discourse has been represented in first-order logic.
Consistency checking task:

Consistency checking task in first-order logic

Let ϕ be the (first-order) semantic representation of the latest sentence in some ongoing discourse, and suppose that the relevant lexical knowledge, world knowledge, natural language metaphysical assumption, and the information from the previous discourse has been represented in first-order logic. Consistency checking task:

$$\mathsf{Lexical} \cup \mathsf{World} \cup \mathsf{NL\text{-}Metaphysical} \cup \mathsf{Discourse\text{-}So\text{-}Far} \models$$

Consistency checking task in first-order logic

Let ϕ be the (first-order) semantic representation of the latest sentence in some ongoing discourse, and suppose that the relevant lexical knowledge, world knowledge, natural language metaphysical assumption, and the information from the previous discourse has been represented in first-order logic.
Consistency checking task:

$$\mathsf{Lexical} \cup \mathsf{World} \cup \mathsf{NL\text{-}Metaphysical} \cup \mathsf{Discourse\text{-}So\text{-}Far} \models \neg\phi$$

We can simplify this a little

$$\text{All-Our-Background-Stuff} \models \neg\phi$$

We can simplify this a little

$$\text{All-Our-Background-Stuff} \models \neg\phi$$

iff

$$\models \text{All-Our-Background-Stuff} \rightarrow \neg\phi$$

We can simplify this a little

$$\text{All-Our-Background-Stuff} \models \neg\phi$$

iff

$$\models \text{All-Our-Background-Stuff} \rightarrow \neg\phi$$

Why? Deduction Theorem!

We can simplify this a little

$$\text{All-Our-Background-Stuff} \models \neg\phi$$

iff

$$\models \text{All-Our-Background-Stuff} \rightarrow \neg\phi$$

Why? Deduction Theorem!

Consequence: we can reduce the consistency checking task to *deciding the validity of a single formula.*

Informativity checking task in first-order logic

Informativity checking task in first-order logic

Let ϕ be the (first-order) semantic representation of the latest sentence in some ongoing discourse, and suppose that the relevant lexical knowledge, world knowledge, natural language metaphysical assumption, and the information from the previous discourse has been represented in first-order logic.

Informativity checking task in first-order logic

Let ϕ be the (first-order) semantic representation of the latest sentence in some ongoing discourse, and suppose that the relevant lexical knowledge, world knowledge, natural language metaphysical assumption, and the information from the previous discourse has been represented in first-order logic.
Informativity checking task:

Informativity checking task in first-order logic

Let ϕ be the (first-order) semantic representation of the latest sentence in some ongoing discourse, and suppose that the relevant lexical knowledge, world knowledge, natural language metaphysical assumption, and the information from the previous discourse has been represented in first-order logic. Informativity checking task:

$$\mathsf{Lexical} \cup \mathsf{World} \cup \mathsf{NL\text{-}Metaphysical} \cup \mathsf{Discourse\text{-}So\text{-}Far} \models$$

Informativity checking task in first-order logic

Let ϕ be the (first-order) semantic representation of the latest sentence in some ongoing discourse, and suppose that the relevant lexical knowledge, world knowledge, natural language metaphysical assumption, and the information from the previous discourse has been represented in first-order logic. Informativity checking task:

$$\mathsf{Lexical} \cup \mathsf{World} \cup \mathsf{NL\text{-}Metaphysical} \cup \mathsf{Discourse\text{-}So\text{-}Far} \models \phi$$

We can also simplify this a little

$$\mathsf{All\text{-}Our\text{-}Background\text{-}Stuff} \models \phi$$

iff

$$\models \mathsf{All\text{-}Our\text{-}Background\text{-}Stuff} \rightarrow \phi$$

Yes - the Deduction Theorem again!

Consequence: we can also reduce this task to *deciding the validity of a single formula.*

So, it's pretty simple, right?

So, it's pretty simple, right?

Um . . . NO! It's about as hard as it gets, actually.

- First (as we shall see) validity is not defined in a way that leads naturally to computation. We need to have another handle on it. That is, we need to open a doorway from model theory to proof theory.
- But even after we've done this, a bigger problem faces us: it turn out we're dealing with an undecidable problem.

Basic inference tasks about deciding validity

As we have just discussed, our two basic inference taks boil down to deciding the validity of certain formulas. In particular:

- Consistency checking: $\models \mathsf{All\text{-}Our\text{-}Background\text{-}Stuff} \rightarrow \neg\phi$
- Informativity checking: $\models \mathsf{All\text{-}Our\text{-}Background\text{-}Stuff} \rightarrow \phi$

Very nice — but . . .

Note:

- This definition is *semantic*.
- That is, it is given in terms of models.

Very nice — but . . .

Note:

- This definition is *semantic*.
- That is, it is given in terms of models.

This is nice in one way (namely, it makes good sense!) but:

- But it is *very* abstract.
- It is defined in terms of *all* models — and there are a lot of models, and most of them are very large.
- So is it of any computational interest whosoever?

Proof theory

- Proof theory is the *syntactic* approach to logic.
- It attempts to define collections of rules and/or axioms that enable us to generate new formulas from old. That is, it attempts to pin down the notion of inference syntactically.
- Given some proof system P, we write $\vdash_P \phi$ to indicate that a formula ϕ is provable in the the proof system. (Incidentally, $\nvdash_P \phi$ means that ϕ is *not* provable in proof system P.).

Many types of proof system

- Natural deduction
- Hilbert-style system (often called axiomatic systems)
- Sequent calculus
- Tableaux systems
- Resolution

Why so many different proof systems?

- Well, one of the most important may simply be that logicians love to play with such systems — and every logician has his or her own favourite pet system!
- A more serious reason is: different proof systems are typically good for different purposes.
- In particular, some systems (notably tableau and resolution) are particularly suitable for computational purposes.

But what does all this have to do with semantics and inference?

- Note: *nothing* we have said so far makes any connection with the model theoretic ideas previously introduced.
- All we have done is talk about provability and vaguely said that we want to "generate" formulas syntactically. What does this have to do with the previous discussion?

But what does all this have to do with semantics and inference?

- Note: *nothing* we have said so far makes any connection with the model theoretic ideas previously introduced.
- All we have done is talk about provability and vaguely said that we want to "generate" formulas syntactically. What does this have to do with the previous discussion?
- Answer: we insist on working with proof systems with two special properties, namely soundness and completeness.

Soundness

- Recall that we write $\models \phi$ to indicate that the formula ϕ is valid (that is, satisfied in all models under all assignments).
- Recall that we write $\vdash_P \phi$ to indicate that ϕ is provable in proof system P.
- We say that a proof system P is sound if and only if

$$\vdash_P \phi \text{ implies } \models \phi$$

.

Explanation

- That is, soundness means that syntactic provability implies semantic validity.
- To put it another way: P does not produce garbage.
- And another: P is "safe".
- Needless to say, all the standard proof systems are sound.

Remark

- Soundness is typically an easy property to prove.
- Proofs typically have some kind of inductive structure. One shows that if the first part of proof is true in a model, then the rules only let us generate formulas that are also true in a model.

Completeness

- Recall that we write $\models \phi$ to indicate that the formula ϕ is valid (that is, satisfied in all models under all assignments).
- Recall that we write $\vdash_P \phi$ to indicate that ϕ is provable in proof system P.
- We say that a proof system p is complete if and only if

$$\models \phi \text{ implies } \vdash_P \phi$$

.

Explanation

- That is, completeness means that our proof system is strong enough to prove everything that is provable.
- To put it another way: if some formula really is true in all models, then our proof system P really is powerful enough to generate it.
- And another: no valid formula is out of reach of our proof system.
- The standard proof systems are complete.

Remark

- Completeness is a much deeper property that soundness, and is a lot more difficult to proof.
- It is typically proved by contraposition. We show that if some formula is not provable ($\nvdash \phi$) then ϕ is *not* valid ($\not\models \phi$). This is done by building a model for $\neg\phi$.
- The first completeness proof for a first-order proof system was given by Kurt Gödel in his 1930 PhD thesis.

Soundness and completeness together

- Recall: proof system P is sound if and only if

$$\vdash_P \phi \text{ implies } \models \phi$$

.

- Proof system P is complete if and only if

$$\models \phi \text{ implies } \vdash_P \phi$$

.

- So if a proof system is both sound and complete (which is what we want) we have that:

$$\models \phi \text{ if and only if } \vdash_P \phi$$

.

- That is, syntactic provability and semantic validity coincide. Sound and complete proof system, really capture the our semantic reality. Working with such systems is *not* just playing with symbols.

So we have made progress

- We now have both a syntactic and a semantic perspective on logic.
- The semantic (model theoretic) perspective guides us conceptually.
- The syntactic (proof theoretical) persepctive gives us a practical, symbol manipulation approach to inference.
- But there is a problem ...

Deciding first-order validity is an Undecidable task

- Deciding validity (in first-order logic) is undecidable.
- That is, no algorithm exists for solving first-order validity.
- Implementing our proof methods for first-order logic (that is, writing a theorem prover only gives us a semi-decision procedure. If a formulas ϕ is valid, the prover will (in principle) be able to prove it..
- But if ϕ is not valid, the prover may never halt!
- So what do we do? Make the best of things! Implement theorem provers, but also implement a partial converse tool: model builders.

Computational Tools

Theorem provers: A tool that, when given a first-order formula ϕ, attempts to prove ϕ. If ϕ is in fact provable a (sound and complete) first-order prover can (in principle) prove it.

Model builders: a tool that, when given a first-order formula ϕ, attempts to build a model for it. It cannot (even in principle) always succeed in this task, but it can be very useful.

Using Theoren Provers and Model Builders

Theorem provers: a mature technology which provides a *negative* check on consistency and informativity — that is, theorem provers can tell us when something is *not* consistent, or *not* informative.

Model builders: a newer technology which provides a (partial) *positive* check on consistency and informativity — that is, model builders can tell us when something is consistent or informative.

What model builders and theorem provers give us

Let BACKGROUND be all our background knowledge, and ϕ the representation of the latest sentence:

What model builders and theorem provers give us

Let BACKGROUND be all our background knowledge, and ϕ the representation of the latest sentence:

- Partial positive test for consistency: give the model builder BACKGROUND $\wedge\, \phi$.

What model builders and theorem provers give us

Let BACKGROUND be all our background knowledge, and ϕ the representation of the latest sentence:

- Partial positive test for consistency: give the model builder BACKGROUND $\wedge \phi$.
- Partial positive test for informativity: give the model builder BACKGROUND $\wedge \neg\phi$.

What model builders and theorem provers give us

Let BACKGROUND be all our background knowledge, and ϕ the representation of the latest sentence:

- Partial positive test for consistency: give the model builder BACKGROUND $\wedge \phi$.
- Partial positive test for informativity: give the model builder BACKGROUND $\wedge \neg\phi$.
- Negative test for consistency: give the theorem prover BACKGROUND $\rightarrow \neg\phi$.

What model builders and theorem provers give us

Let BACKGROUND be all our background knowledge, and ϕ the representation of the latest sentence:

- Partial positive test for consistency: give the model builder BACKGROUND $\wedge \phi$.
- Partial positive test for informativity: give the model builder BACKGROUND $\wedge \neg\phi$.
- Negative test for consistency: give the theorem prover BACKGROUND $\rightarrow \neg\phi$.
- Negative test for informativity: give the theorem prover BACKGROUND $\rightarrow \phi$.

What model builders and theorem provers give us

Let BACKGROUND be all our background knowledge, and ϕ the representation of the latest sentence:

- Partial positive test for consistency: give the model builder BACKGROUND $\wedge \phi$.
- Partial positive test for informativity: give the model builder BACKGROUND $\wedge \neg\phi$.
- Negative test for consistency: give the theorem prover BACKGROUND $\rightarrow \neg\phi$.
- Negative test for informativity: give the theorem prover BACKGROUND $\rightarrow \phi$.

And do this in parallel using the best available software!

Putting it together: the CURT programs

CURT

(Clever Use of Reasoning Tools)

- Baby Curt No inference capabilities
- Rugrat Curt: negative consistency checks (naive prover)
- Clever Curt: negative consistency checks (sophisticated prover)
- Sensitive Curt: negative and positive informativity checks
- Scrupulous Curt: eliminating superfluous readings
- Knowledgeable Curt: adding background knowledge
- Helpful Curt: question answering

Baby Curt computes semantic representations. . .

Baby Curt computes semantic representations...

```
Curt:  'Want to tell me something?'
```

Baby Curt computes semantic representations...

```
Curt:  'Want to tell me something?'
> every boxer loves a woman
```

Baby Curt computes semantic representations...

```
Curt:  'Want to tell me something?'
> every boxer loves a woman
Curt:  'OK.'
```

Baby Curt computes semantic representations...

```
Curt:  'Want to tell me something?'
> every boxer loves a woman
Curt:  'OK.'
> readings
```

Baby Curt computes semantic representations...

```
Curt:  'Want to tell me something?'
> every boxer loves a woman
Curt:  'OK.'
> readings
1 forall A (boxer(A) > exists B (woman(B) & love(A, B)))
2 exists A (woman(A) & forall B (boxer(B) > love(B, A)))
```

Baby Curt accumulates information . . .

Baby Curt accumulates information . . .

```
> mia walks
```

Baby Curt accumulates information . . .

```
> mia walks
Curt:  'OK.'
```

Baby Curt accumulates information . . .

```
> mia walks
Curt:  'OK.'
> vincent dances
```

Baby Curt accumulates information . . .

```
> mia walks
Curt:  'OK.'
> vincent dances
Curt:  'OK.'
```

Baby Curt accumulates information . . .

```
> mia walks
Curt:  'OK.'
> vincent dances
Curt:  'OK.'
> readings
```

Baby Curt accumulates information ...

```
> mia walks
Curt:  'OK.'
> vincent dances
Curt:  'OK.'
> readings
1 (walk(mia) & dance(vincent))
```

Alas — Baby Curt is very stupid...

Alas — Baby Curt is very stupid...

```
> mia walks
```

Alas — Baby Curt is very stupid...

```
> mia walks
Curt:  'OK.'
```

Alas — Baby Curt is very stupid...

```
> mia walks
Curt:  'OK.'
> mia does not walk
```

Alas — Baby Curt is very stupid...

```
> mia walks
Curt:  'OK.'
> mia does not walk
Curt:  'OK.'
```

Alas — Baby Curt is very stupid...

```
> mia walks
Curt:  'OK.'
> mia does not walk
Curt:  'OK.'
> ?- readings 1 (walk(mia) & - walk(mia))
```

Adding an inference component

- Key idea — use sophisticated theorem provers and model builders in parallel.
- The theorem prover provides negative check for consistency and informativity.
- The model builder provides positive check for consistency and informativity.
- The first to find a result, reports back, and stops the other check.

Example

Example

```
> Vincent is a man
```

Example

```
> Vincent is a man

Message (consistency checking):  mace found a result.
Curt:  OK.
```

Example

```
> Vincent is a man

Message (consistency checking):  mace found a result.
Curt:  OK.

> ?- models
```

Example

```
> Vincent is a man

Message (consistency checking):  mace found a result.
Curt:  OK.

> ?- models

1 model([d1], [f(1, man, [d1]), f(0, vincent, d1)])
```

Example (continued)

Example (continued)

```
> Mia likes every man.
```

Example (continued)

```
> Mia likes every man.
Message (consistency checking):  mace found a result.
Curt:  OK.
```

Example (continued)

```
> Mia likes every man.

Message (consistency checking):  mace found a result.
Curt:  OK.

> Mia does not like Vincent.
```

Example (continued)

```
> Mia likes every man.

Message (consistency checking):  mace found a result.
Curt:  OK.

> Mia does not like Vincent.

Message (consistency checking):  bliksem found a
result.
Curt:  No!  I do not believe that!
```

Background knowledge

- Because we are working with first-order logic, it is straightforward to add background knowledge.
- Knowledgeable Curt has a information about antonyms, hypernyms, gender characteristics of proper names, and so on, stored in its lexicon. When Knowledgeable Curt is invoked, this information is automatically compiled into first-order logic.
- Knowledgeable Curt also has a (very) small fund of world knowledge at its disposal.

Another example

Another example

```
> ?- every car has a radio
```

Another example

```
> ?- every car has a radio
Message (consistency checking):  mace found a result.
Message (consistency checking):  bliksem found a
result.
Curt:  'OK.'
```

Another example

```
> ?- every car has a radio
Message (consistency checking):  mace found a result.
Message (consistency checking):  bliksem found a
result.
Curt:  'OK.'
> ?- readings
```

Another example

```
> ?- every car has a radio

Message (consistency checking):  mace found a result.
Message (consistency checking):  bliksem found a
result.
Curt:  'OK.'

> ?- readings

1 forall A (car(A) > exists B (radio(B) & have(A,
B)))
```

How Curt worked this out

```
(forall A (concrete(A) >   abstract(A)) &
(forall B (entity(B) > concrete(B)) &
(forall C (entity(C) > thing(C)) &
(forall D (artifact(D) > inedible(D)) &
(forall E (artifact(E) > object(E)) &
(forall F (mobile(F) >   immobile(F)) &
(forall G (vehicle(G) > instrument(G)) &
(forall H (instrument(H) > mobile(H)) &
(forall I (instrument(I) > artifact(I)) &
(forall J (object(J) > nonliving(J)) &
(forall K (object(K) > entity(K)) &
```

And much else besides . . . it's not easy computationally!

Some questions

- Is a logic-based approach to feasible? How far can it be pushed?
- Is first-order logic essential?
- Are there other interesting inference tasks?
- Is any of this relevant to current trends in computational linguistics, where shallow processing and statistical approaches rule?
- Are there applications?

Reminder: a useful URLS

- `www.blackburnbos.org`. Contains lots of material relating to the "Representation and Inference" book, including all the Prolog code for the book; including DCGs, lambda software, scoping software, and the CURT programs.