Discourse

BSc Artificial Intelligence, Spring 2011

Raquel Fernández

Institute for Logic, Language & Computation
University of Amsterdam

Summary from Last Week

We looked into the following two tasks:

- Consistency Checking Task: given the logical representation φ of a discourse, is φ consistent (i.e. satisfiable) or inconsistent (i.e. unsatisfiable)?
  - φ is satisfiable if it is satisfied at least in one model
- Informativity Checking Task: given the logical representation φ of a discourse, is φ informative (i.e. invalid) or uninformative (i.e. valid)?
  - φ is valid if it is satisfied in all models under any variable assignment

These tasks require us to check all possible models, but there is an infinite number of possible models! Therefore there is no algorithm capable of solving these tasks in finite time for all possible input formulas... These tasks are *undecidable* for FOL.

However, automated reasoning tools such as theorem provers and model builders give us a partial computational solution.

Automated Reasoning Tools

A theorem prover is a system that systematically checks whether a formula is valid using refutation proof methods:

- to show that φ is valid, the prover attempts to show that $\neg\varphi$ leads to a contradiction
- if a proof is found, φ is valid; if a proof is not found, we are left wondering whether a proof doesn't exists or it will be found later on...
- theorem provers cannot prove non-validity.

A model builder is a system that, given a formula, attempts to build a model that satisfies it:

- a model builder can thus prove satisfiability
- model builders are only capable to build relatively small finite models;
- they do not build arbitrary infinite models, so even when some possible models exist they may not be able to always build them.

How can we use these tools for consistency & informativity checking?

Consistency Checking

Everybody likes Mia. She's smart and she dances gorgeously. Mia likes Butch.
Butch is a boxer ($\rightsquigarrow$ *consistent*) / Butch doesn't like Mia ($\rightsquigarrow$ *inconsistent*)

- ψ is consistent with respect to $\varphi_1, \ldots, \varphi_n$ iff
  - $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \psi$ is satisfiable
    there is at least one model where the formula is true
  - $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \neg\psi$ is invalid
    $\neg\psi$ is not true in all models where $\varphi_1 \wedge \ldots \wedge \varphi_n$ is true
- ψ is inconsistent with respect to $\varphi_1, \ldots, \varphi_n$ iff
  - $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \psi$ is unsatisfiable
    there are no models where the formula is true
  - $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \neg\psi$ is valid
    $\neg\psi$ is true in all models where $\varphi_1 \wedge \ldots \wedge \varphi_n$ is true

- We give $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \neg\psi$ to a theorem prover, which attempts to prove validity by refuting $\neg[\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \neg\psi]$. If a proof is found, the original formula is valid and hence the discourse is inconsistent.
- We give $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \psi$ to a model builder. If it finds a model for it, the formula is satisfiable and hence the discourse is consistent.

Informativity Checking

Everybody likes Mia. She's smart and she dances gorgeously. Mia likes Butch.
Butch is a boxer ($\rightsquigarrow$ *informative*) / Mia is smart ($\rightsquigarrow$ *uninformative*)

- ψ is informative with respect to $\varphi_1, \ldots, \varphi_n$ iff
  - $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \psi$ is invalid
    the formula is not true in all models
  - $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \neg\psi$ is satisfiable
    there is at least one model where $\varphi_1 \wedge \ldots \wedge \varphi_n$ and $\neg\psi$ are true
- ψ is uninformative with respect to $\varphi_1, \ldots, \varphi_n$ iff
  - $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \psi$ is valid
    it is true in all models
  - $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \neg\psi$ is unsatisfiable
    there are no models where the formula is true

- We give $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \psi$ to a theorem prover, which attempts to prove validity by refuting $\neg[\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \psi]$. If a proof is found, the original formula is valid and hence the discourse is uninformative.
- We give $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \neg\psi$ to a model builder. If it finds a model for it, the formula is satisfiable and hence the discourse is informative.

Combining both Tools

We have seen that:

- Theorem provers are capable of carrying out negative checks for consistency and informativity.
- Model builders are capable of carrying out partial positive checks for these tasks.

The optimal computational strategy is thus to use simultaneously theorem proving and model building on the input formula – that is, to do both negative and (partial) positive checks in parallel.

The architecture developed by Blackburn & Bos deals with the consistency and informativity tasks by carrying out these processes in parallel.

Some Technicalities

We shall use the theorem prover Prover9 and the model builder Mace4. Both can be downloaded from `http://www.cs.unm.edu/~mccune/prover9/`

Prover9 and Mace4 are available in all FNWI Linux machines. To activate them, you need to do the following:

- Log in with your student account.
- In your home directory you'll find the file `.pkgrc`. This is a hidden file; if you can't see it, tick the option "Show Hidden Files" under "View".
- Open `.pkgrc` with a text editor, add `ladr` at the bottom of the file, and save it.
- Open a terminal, run the command `eval 'softpkg'` and kill the terminal.

Prover9 and Mace4 are newer versions of the reasoning tools originally employed by B&B. Updated version of the B&B Prolog code for use with these newer tools are packed in `updated-inference.zip`, which can be downloaded from the course materials in Blackboard.

Inference Architecture (1)

Files in `updated-inference` (adapted from BB1):

- `callInference.pl`: main Prolog interface to off-the-shelf theorem provers or model builders. It calls all other programs.
- `inferenceEngines.pl`: specifies the inference engines to be used.
- `fol2prover9.pl`: translates a FOL formula to Prover9/Mace4 syntax.
- `interfaceTP.perl`: Perl script interfacing theorem provers.
- `interfaceMB.perl`: Perl script interfacing model builders.
- `interfaceTPandMB.perl`: Perl script interfacing theorem provers and model builders.

Inference Architecture (2)

The program `callInference.pl` includes the following predicates:

- The predicates `fol2prover9/2` and `fol2mace4/2` translate a formula in Prolog notation to instructions for Prover9 & Mace4.

Try the following:

```
?- fol2prover9(imp(all(X,dance(X)),not(some(Y,not(dance(Y))))),user).
?- fol2mace4(some(X,and(man(X),all(Y,imp(woman(Y),love(X,Y))))),user).
```

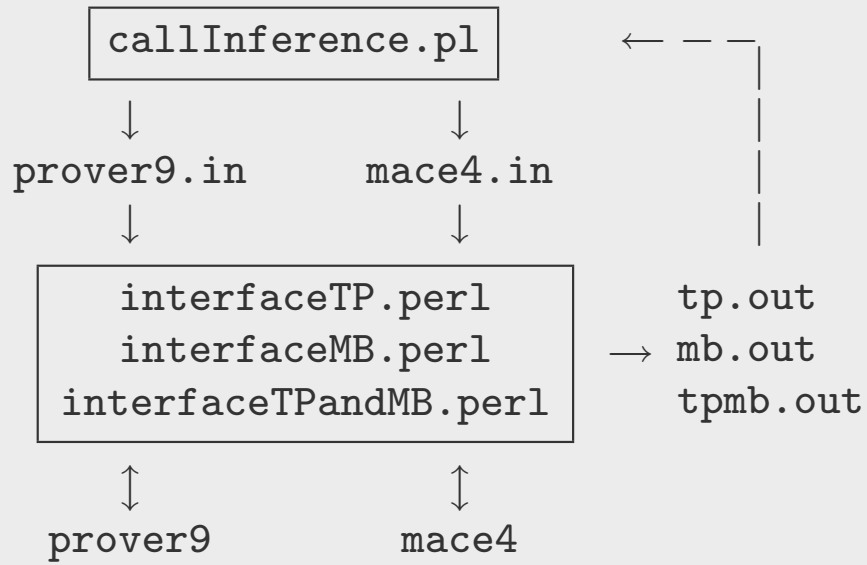
[N.B: with `user` as second argument, Prolog prints the result to standard output.]

- The interface predicates `callTP(Problem,Result,Prover)`, `callMB(Problem,DomainSize,Model,ModelBuilder)`, and `callTPandMB(TPProblem,MBProblem,DSize,Proof,Model,Engine)`

Try the following:

```
?- callTP(imp(all(X,dance(X)),not(some(Y,not(dance(Y))))),R,P).
?- callMB(some(X,and(man(X),all(Y,imp(woman(Y),love(X,Y))))),2,M,MB).
```

Inference Architecture (3)



- `callInference.pl` uses `fol2prover9/2` and `fol2mace4/2` to create input files for the engines.
- The Perl scripts act as interfaces:
  - they activate the engines with the created input files,
  - read their output and convert it to the right notation,
  - write the result in an output file
- The output file is then given back to `callInference.pl`.

Exercises

A sheet with exercises for Practical Session #1 can be downloaded from the course materials in Blackboard.