Context Theorem Proving

by Sven Nilsen, 2018

Non-monotonic logic is used to infer consequences that can be retracted based on further evidence. Here I represent a version of non-monotonic logic that can be simulated in a monotonic solver, making it possible to compile non-monotonic problems to formats that are accepted by existing automated theorem provers.

For example, consider the following:

$$(a \wedge b) \wedge (a = b)$$

It is known that:

$$(a \wedge b) \rightarrow (a = b)$$

Therefore:

$$((a \wedge b) \wedge (a = b)) = (a \wedge b)$$

A more general version:

$$((x \wedge y) \wedge (x \rightarrow y)) \rightarrow ((x \wedge y) = x)$$

There are many cases where implication `x → y` holds for a common case, which is something desirable to express in the proof. A trick to achieve this is by introducing a `ut` variable that stands for "usually true". For every assumption, instead of assuming it is true, the truth value of the assumption is set equal to `ut`. Then, `ut` is added as a new assumption, which allows proving the same conclusions as before.

$$(((x \wedge y) = ut) \wedge ((x \rightarrow y) = ut) \wedge ut) \rightarrow ((x \wedge y) = x)$$

For example, if `ut` means "usually true for birds", one can prove that flying ability is assumed when talking about birds.

$$(((bird \rightarrow fly) = ut) \wedge ut) \rightarrow (bird \rightarrow fly)$$

The non-monotonic problem is compiled this way to a monotonic problem and fed to a monotonic solver. Whenever the solver reaches a wrong conclusion, the `ut` assumption is changed `¬ut` and the problem is attempted resolved. If the conclusion is right this time, the conclusion is accepted, under the condition that this is not usual.

Alternatively, one can leave out the `ut` assumption and use it in the conclusion in the form:

$$\langle assumption \rangle \rightarrow (ut \rightarrow \langle conclusion\ given\ context\ `ut` \rangle)$$

There can be more than one `ut` variable, named `$ut_0$`, `$ut_1$` etc. These variables maps to different contexts. In such cases, `¬ut` is not allowed. Any conclusion must hold for at least one `$ut_x$` variable. One can reason about which contexts that various objects are used.

For example, in one context, a bird can fly, but in another context, a bird can not fly:

$$(\text{bird} \rightarrow \text{fly}) = ut_0$$
$$(\text{bird} \rightarrow \neg\text{fly}) = ut_1$$

Since `fly` can only be true or false, the following must be true:

$$ut_0 \vee ut_1$$

There is no context where a bird either can fly or does not fly. This holds even when adding new assumptions. Because of this non-trivial nature of constraints between various contexts, one can never reason consistently by assuming a context does not hold, or that two contexts holds at the same time, because this can lead to a false assumption. However, it is possible to reason consistently when at least one of possible contexts holds:

$$(ut_0 \vee ut_1) \rightarrow ...$$

Therefore, the logical OR operator is the only valid one for contexts.