Logical reasoning and programming, task V

(January 5, 2018)

Problem

We have seen that it is easy to add new Boolean connectives, e.g. equivalence, to our systems, because all Boolean connectives are definable there. However, sometimes we want to add more complex connectives and operators. This leads to so called non-classical logics. For example, we can add a unary modal operator *necessarily*, usually denoted $\Box$ (called box), where $\Box p$ means "necessarily p". Logics containing such operators are called modal logics and were studied already by Aristotle. They occur quite frequently in computer science, e.g., in temporal and description logics.

Clearly, we should define the meaning of box, but we do not provide it. Instead of that, we shall present a tableau system for the weakest normal modal propositional logic, called K, and define a provability in K using this system. Your task is to implement this tableau system in Prolog. Although you are encouraged to find some more details about K, this is not necessary for completing the task. Let us also note that the tableau system presented here differs slightly from the tableau system presented at lectures.

We are in propositional logic and our language contains only propositional variables (e.g., p), conjunction (e.g., $p \wedge q$), negation (e.g., $\neg p$), and the operator necessarily (e.g., $\Box p$). Formulae are then defined in the standard way.

Our tableau system for K deals with the sets of formulae. A formula φ is provable in K iff we can derive that $\{\neg\varphi\}$ is inconsistent using the tableau system for K. The tableau system contains rules that preserve inconsistency, hence if a set of formulae (above line) is inconsistent, then another set of formulae (below line) is inconsistent as well. It consists of the following rules:

$$\frac{\bot}{\Gamma \cup \{\varphi, \neg\varphi\}}\ (\bot) \qquad \frac{\Gamma \cup \{\varphi\}}{\Gamma \cup \{\neg\neg\varphi\}}\ (\neg)$$

$$\frac{\Gamma \cup \{\varphi, \psi\}}{\Gamma \cup \{\varphi \wedge \psi)\}}\ (\wedge) \qquad \frac{\Gamma \cup \{\neg\varphi\} \qquad \Gamma \cup \{\neg\psi\}}{\Gamma \cup \{\neg(\varphi \wedge \psi)\}}\ (\vee)$$

$$\frac{\Gamma}{\Gamma \cup \Delta}\ (\subset) \qquad \frac{\Gamma \cup \{\neg\varphi\}}{\Box\Gamma \cup \{\neg\Box\varphi\}}\ (\mathsf{K})$$

where Γ and Δ are sets of formulae, φ and ψ are formulae, and $\Box\Gamma = \{\, \Box\varphi \mid \varphi \in \Gamma \,\}$. The special symbol $\bot$ represents inconsistency and the rule $(\bot)$ says that a set of formulae containing φ and $\neg\varphi$ is trivially inconsistent, we also say that $(\bot)$ closes a branch. Only the rule $(\vee)$ contains branching — to prove that $\Gamma \cup \{\neg(\varphi \wedge \psi)\}$ is inconsistent we have to prove that both $\Gamma \cup \{\neg\varphi\}$ and $\Gamma \cup \{\neg\psi\}$ are inconsistent.

A set of formulae is inconsistent if there is a derivation of this set using the given rules such that all branches are closed (start with $\bot$).

The following example is a proof of formula $\neg(\Box p) \wedge ((\Box\neg(p \wedge \neg q)) \wedge (\neg\Box q))$, which is equivalent to $\Box p \to (\Box(p \to q) \to \Box q)$ in the language with the implication. In other words, it says "if necessarily p and necessarily

$p \to q$, then necessarily q''. A basic principle which is valid in all normal modal logics.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\bot}{\{p, \neg p, \neg q\}}\,(\bot) \quad \dfrac{\bot}{\{p, \neg\neg q, \neg q\}}\,(\bot)}{\{p, \neg(p \wedge \neg q), \neg q)\}}\,(\vee)}{\{\Box p, \Box\neg(p \wedge \neg q), \neg\Box q\}}\,(\mathsf{K})}{\{\Box p, (\Box\neg(p \wedge \neg q)) \wedge (\neg\Box q)\}}\,(\wedge)}{\{(\Box p) \wedge ((\Box\neg(p \wedge \neg q)) \wedge (\neg\Box q))\}}\,(\wedge)}{\{\neg\neg((\Box p) \wedge ((\Box\neg(p \wedge \neg q)) \wedge (\neg\Box q)))\}}\,(\neg)$$

The depth of such a proof, which is clearly a binary tree, is 6, because the longest path from $\bot$ to the root contains six applications of rules. Hence we say that $\neg(\Box p) \wedge ((\Box\neg(p \wedge \neg q)) \wedge (\neg\Box q))$ has a proof of depth 6.

For further details about tableau systems for modal logics see, e.g., this text.

Program

The following representation of formulae in Prolog is used

Logic	Prolog
p	`p`
$\neg p$	`-(p)`
$p \wedge q$	`(p,q)`
$\Box p$	`box(p)`

and it naturally extends to all formulae. Propositional variables are atoms in Prolog, a conjunctions is a pair, a negation is expressed by a unary minus, and a box operator is expressed by a unary function `box`.

For example, $\neg\neg((\Box p) \wedge ((\Box\neg(p \wedge \neg q)) \wedge (\neg\Box q)))$ is represented in Prolog by `-(-((box(p),(box(-((p,-(q)))),-(box(q))))))`.

You are supposed to upload a program `modaltap.pl`, in an archive, containing a binary predicate `prove`, where the first argument is an input formula and the second argument is a number. Hence we have `prove(-Formula, -Depth)` and it succeeds iff `Formula` has a proof of depth at most `Depth`.

For example, `prove(-((box(p),(box(-((p,-(q)))),-(box(q))))), 6).` succeeds and it also succeeds for the second argument being $7, 8, \ldots$.

Tips

It is wise to represent sets of formulae using lists. In this case you can use the predicate `select/3`, see help for further details.

There is no need to optimize your program for efficiency, your task is to write a simple program. A strightforward implementation of the presented tableau system should work just fine.