LoTREC: Theory and Practice

B. Saïd
O. Gasquet, F. Schwarzentruber, A. Herzig

Institut de Recherche en Informatique de Toulouse - IRIT
Université Paul Sabatier - UPS

Tableaux 2009
Outline

Part 1: Theory
1. Modal logics
2. Reasoning

Part 2: Practice
3. LoTREC
4. Implementation
Part 1: Theory

1. Modal logics
   - Possible worlds models
   - Constraints on models
   - Syntax
   - Semantics

2. Reasoning
   - Classes of modal logics
   - Reasoning in a class of modal logic
   - Automated reasoning
What is a model?

- Possible worlds
  - node
  - states

- Valuation
  - labeling function
  - interpretation

- Accessibility relation
  - labeled edges
  - transitions

- Model
  - labeled graph
  - transition system
Kripke Model

Given two disjoint sets of symbols $\mathcal{P}$ and $\mathcal{I}$

- $M = (W, R, V)$
  - $W$ non-empty set of possible worlds
  - $R: \mathcal{I} \rightarrow 2^{W \times W}$ an accessibility relation
  - $V: W \rightarrow 2^\mathcal{P}$ a valuation function

- Pointed model $M, w$
  where $w \in W$ is the actual world
Readings of “$R$”

- **Alethic**
  \[ wRu \text{ iff } u \text{ is possible given the actual world } w \]

- **Temporal**
  \[ wRu \text{ iff } u \text{ is in the future of } w \]

- **Epistemic**
  \[ wR_iu \text{ iff } u \text{ is possible for agent } i, \text{ given the actual world } w \]

- **Dynamic**
  \[ wR_Pu \text{ iff } u \text{ is a possible result of the execution of the program / action } P \text{ in } w \]

Readings of $R \Rightarrow$ Properties of $R$
Readings of “$R$”

- **Alethic**
  \[ wRu \iff u \text{ is possible given the actual world } w \]

- **Temporal**
  \[ wRu \iff u \text{ is in the future of } w \]

- **Epistemic**
  \[ wR_i u \iff u \text{ is possible for agent } I, \text{ given the actual world } w \]

- **Dynamic**
  \[ wR_P u \iff u \text{ is a possible result of the execution of the program / action } P \text{ in } w \]

Readings of $R \Rightarrow$ Properties of $R$
Constraints on “$R$”

One relation:

- Serial
  
  *there is always a future*
  
  for all $w$ exists $u$ s.t. $wRu$

- Transitive

  *future of future is future*
  
  I know what I know

- Reflexive

  *I know “smthg” i.e. it is true*

- Symmetric

- Equivalence (universal)

- Confluent (Church-Rosser)

- . . .

Two or more:

- $R_I$ included in $R_J$

- $R_I = R_J \cup R_K$

- $R_J = (R_I)^{-1}$

- $R_J = (R_I)^*$
  
  (transitive closure)

- $R_I \circ R_J = R_J \circ R_I$

- Confluent

- . . .
Constraints on “V”

- Nominal: is unique
  if $N \in V(w)$ and $N \in V(u)$ then $w = u$

- Intuitionistic: atomic propositions are persistent
  for every atom $P$ and world $w$:
  if $P \in V(w)$ and $wRu$ then $P \in V(u)$

- ...
Modal operators

- Express **non-truth functional** concepts: belief, time, action, obligation, knowledge, conditional...

- Schema $op(a_1, \ldots, a_n)$
  - $Bel_I A$: agent $I$ believes that $A$
  - $FA$: $A$ will be true at some point in the future
  - $After_P A$: $A$ will be true after the execution of $P$

- ... 

- Generic form
  - $\Box A$: $A$ is **necessary** (true in all possible worlds)
  - $\Diamond A$: $A$ is **possible**

- In general:
  - $\Diamond A \iff \neg \Box \neg A$
  - $\Box A \iff \neg \Diamond \neg A$
What is a formula?

BNF:

\[ A ::= P | \text{op}(A_1, \ldots, A_n) \]

i.e.

\[ A ::= P | \neg A | A \land A | A \lor A | \diamond A | \Box A | \langle A \rangle A | [A]A | K_A A \ldots \]

Example

- \[ P \land \neg Q \land \Box Q \land \diamond (P \land \Box \neg Q) \]
- \[ Door\_\text{Closed} \land [\text{Open}] Door\_\text{Open} \]
- \[ Card\_\text{Is\_Red} \land K_{Ann} Card\_\text{Is\_Red} \land K_{Ann} \neg K_{Bob} Card\_\text{Is\_Red} \]
Truth conditions

- **Atoms**
  - $M, w \models P$ iff $P \in V(w)$

- **Classical connectives**
  - $M, w \models A \land B$ iff $M, w \models A$ and $M, w \models B$
  - ...  

- **Non-classical operators**
  - Via the accessibility relation $R$
    - $M, w \models \Diamond A$ iff exists $u$: $wRu$ and $M, u \models A$
    - $M, w \models \square A$ iff for all $u$: if $wRu$ then $M, w \models A$
Truth conditions

- Multi-modal operators
  - $M, w \models \langle I \rangle A$ iff exists $u$: $wR_i u$ and $M, u \not\models A$
  - ... 

- Algebraic operators
  - $M, w \models \Box^{-1} A$ iff exists $u$: $wR^{-1} u$ and $M, u \not\models A$
  - $M, w \models \langle I \cup J \rangle A$ iff exists $u$: $w(R_i \cup R_J) u$ and $M, u \not\models A$
  - $M, w \models \langle I^* \rangle A$ iff exists $u$: $wR^*_i u$ and $M, u \not\models A$
Temporal operators

- \( M, w \models XA \) iff exists \( u: wRu \) and \( M, u \models A \)
- \( M, w \models \diamond A \) iff exists \( n,u: wR^nu \) and \( M, u \models A \)

\[
\begin{array}{c}
A \rightarrow A \rightarrow \ldots \rightarrow A \rightarrow B \\
w \rightarrow u
\end{array}
\]

- \( M, w \models AUB \) iff exists \( u: wR^*u \) and \( M, u \models B \)
and for all \( v: (wR^*v \text{ and } vR^+u), M, v \models A \)

\[
\begin{array}{c}
A \rightarrow \ldots \\
w \rightarrow u
\end{array}
\]
Part 1: Theory

1 Modal logics
   - Possible worlds models
   - Constraints on models
   - Syntax
   - Semantics

2 Reasoning
   - Classes of modal logics
   - Reasoning in a class of modal logic
   - Automated reasoning
Basic modal logic

$K = \text{the set of all models (Kripke)}$

- $A$ is valid in $K$ iff for all $M$ in $K$ and all $w$ in $M$: $M, w \vDash A$

Example

- $\Box (P \lor \neg P)$
- $\Box (P \land Q) \rightarrow \Box P \land \Box Q$
- $\Box P \land \Box Q \rightarrow \Box (P \land Q)$

- $A$ is satisfiable in $K$ iff for some $M$ in $K$ and some $w$ in $M$: $M, w \vDash A$

Example

- $P$
- $P \land \neg \Box P$
- $P \land \Box \neg P$
- $\Box P \land \neg \Box \Box P$
Other Classes

$C = \text{some subset of } K$

- $A$ is valid in $C$ iff for all $M$ in $C$ and all $w$ in $M$: $M, w \vDash A$

**Example**

- $\Box P \rightarrow P$ invalid in $K$
  
  $\Box P, \neg P \rightarrow P$

- $\Box P \rightarrow P$ valid in the class of reflexive models

- $\Diamond P \rightarrow \Diamond \Diamond P$ valid in transitive models

- $A$ is satisfiable in $C$ iff for some $M$ in $C$ and some $w$ in $M$: $M, w \vDash A$

**Example**

- $P \land \Box \neg P$ is satisfiable in $K$

- $P \land \Box \neg P$ is unsatisfiable in reflexive models

$A$ is valid in $C$ iff $\neg A$ is unsatisfiable in $C$
Exemple of classes

- Singleton models \( \{ M : \text{card}(W) = 1 \} \)
  \[ \Diamond A \rightarrow \square A \]

- Reflexive models (KT)
  \[ \square A \rightarrow A \]

- Transitive models (S4)
  \[ \Diamond \Diamond A \rightarrow \Diamond A \]

- Equivalence relation (S5)
  \[ A \rightarrow \square \Diamond A \]

- ...
Reasoning problems

- Model checking
  Given $A$, $M$ and $w$ do we have $M, w \models A$?

- Validity
  Given $A$ and $C$ is $A$ valid in $C$?

- Satisfiability
  Given $A$ and $C$ does there exist $M$ in $C$ and $w$ in $M$: $M, w \models A$?

- Model construction
  Given $A$ and $C$ compute $M$ in $C$ and $w$ in $M$: $M, w \models A$

How can we solve them automatically?
Classical logic [Beth]

Checking the satisfiability of $A_0$

1. Try to find $M$ and $w$ by applying truth conditions (tableaux rules)
   - $M, w \vDash A \land B \implies$ add $M, w \vDash A$ and add $M, w \vDash B$
   - $M, w \vDash A \lor B \implies$ add either $M, w \vDash A$ or add $M, w \vDash B$ (non det.)
   - $M, w \vDash \neg A \implies$ don’t add $M, w \vDash A$ !!
     - $M, w \vDash \neg \neg A \implies$ add $M, w \vDash A$
     - $M, w \vDash \neg (A \lor B) \implies$ add $M, w \vDash \neg A$ and add $M, w \vDash \neg B$
     - $M, w \vDash \neg (A \land B) \implies$ add $M, w \vDash \neg A$ or add $M, w \vDash \neg B$

2. apply while possible (saturation)

3. is $M$ a model?
   - NO if both $M, w \vDash B$ and $M, w \vDash \neg B$ (closed tableau)
   - ELSE $M$ is a model for $A_0$ (open tableau)

$W = \{ w \}$, $R = \emptyset$, $V(w) = \{ P : M, w \vDash P \}$
Classical logic [Beth]

Checking the satisfiability of $A_0$

1. Try to find $M$ and $w$ by applying truth conditions (tableaux rules)
   - $M, w \models A \land B \implies$ add $M, w \not\models A$ and add $M, w \not\models B$
   - $M, w \models A \lor B \implies$ add either $M, w \models A$ or add $M, w \not\models B$ (non det.)
   - $M, w \models \neg A \implies$ don’t add $M, w \models A$ !
     - $M, w \not\models \neg \neg A \implies$ add $M, w \not\models A$
     - $M, w \not\models \neg (A \lor B) \implies$ add $M, w \models \neg A$ and add $M, w \models \neg B$
     - $M, w \not\models \neg (A \land B) \implies$ add $M, w \models \neg A$ or add $M, w \models \neg B$

2. apply while possible (saturation)

3. is $M$ a model?
   - NO if both $M, w \models B$ and $M, w \models \neg B$ (closed tableau)
   - ELSE $M$ is a model for $A_0$ (open tableau)

$W = \{w\}, \quad R = \emptyset, \quad V(w) = \{P : M, w \models P\}$
Modal logic

Basic cases

- $M, w \models \diamond A$
  - $\Rightarrow$ add some new node $u$, add $wRu$, add $M, u \models A$

- $M, w \models \Box A$
  - $\Rightarrow$ for all node $u$ s.t. $wRu$, add $M, u \models A$

Apply truth conditions = Build a labeled graph

- creates nodes
- add links
- add formulas to nodes
Example

a node with the input formula

\[ [] \, P \& \leftrightarrow \, Q \& \leftrightarrow \, (R \lor \sim \, P) \]
Example

\[ M, w \not\models A \land B \text{ iff } M, w \not\models A \text{ and } M, w \not\models B \]

\[ A \text{ is } \Box P \]

\[ B \text{ is } \Diamond Q \land \Diamond (R \lor \neg P) \]

\[ [] P \land \leftrightarrow Q \land \leftrightarrow (R \lor \neg P) \]
Example

\[ M, w \models A \land B \text{ iff } M, w \models A \text{ and } M, w \models B \]

- \( A \) is \( \Box P \)
- \( B \) is \( \Diamond Q \land \Diamond (R \lor \neg P) \)

\[
\begin{align*}
[] P & \land \leftrightarrow Q \land \leftrightarrow (R \lor \neg P) \\
[] P & \\
\leftrightarrow Q & \land \leftrightarrow (R \lor \neg P)
\end{align*}
\]
Example

\[ M, w \models A \land B \iff M, w \models A \text{ and } M, w \models B \]

\[
\begin{align*}
[] P & \land <> Q & <> (R \lor \neg P) \\
[] P & \\
<> Q & <> (R \lor \neg P) \\
<> Q & \\
<> (R \lor \neg P)
\end{align*}
\]
Example

\[ M, w \models \Diamond A \text{ iff } \exists u \mid wRu \text{ and } M, u \models A \]

\[
\begin{array}{c}
\Box P & \land & \diamondsuit Q & \land & \diamondsuit (R \lor \neg P) \\
\Box P \\
\diamondsuit Q & \land & \diamondsuit (R \lor \neg P) \\
\diamondsuit Q \\
\diamondsuit (R \lor \neg P) \\
\end{array}
\]

\[
\begin{array}{c}
R \\
Q \\
\end{array}
\]
Example

\[ M, w \models □A \iff \forall u : wRu \text{ then } M, u \models A \]

\[
\begin{align*}
\□ P & \land \langle\rangle Q \land \langle\rangle (R v \sim P) \\
\□ P & \\
\langle\rangle Q & \land \langle\rangle (R v \sim P) \\
\langle\rangle Q & \\
\langle\rangle (R v \sim P) &
\end{align*}
\]
Example

\[ M, w \models A \lor B \text{ iff } M, w \models A \text{ or } M, w \models B \]

Premodel 1

Premodel 2
Example

[p] P & [<> Q & [<> (R v ~ P)]
[p]
[<> Q & [<> (R v ~ P)]
[<> Q
[<> (R v ~ P)]

premodel 1

[p] P & [<> Q & [<> (R v ~ P)]
[p]
[<> Q & [<> (R v ~ P)]
[<> Q
[<> (R v ~ P)]

premodel 2
Example

### Premodel 1

\[
\begin{align*}
\Box P & \land \langle< Q & \land \langle< (R \lor \neg P) \\
\Box P & \\
\langle< Q & \land \langle< (R \lor \neg P) \\
\langle< Q & \\
\langle< (R \lor \neg P) & 
\end{align*}
\]

\[
\begin{align*}
R & \\
R & \\
R & \\
\end{align*}
\]

\[
\begin{align*}
R \lor \neg P & \\
P & \\
R & \\
\end{align*}
\]

\[
\begin{align*}
Q & \\
P & \\
Q & \\
\end{align*}
\]

### Model

\[
\begin{align*}
\neg P & \lor P \\
\neg Q & \lor Q \\
\neg R & \lor R \\
\end{align*}
\]

### Extraction

\[
M, w \vDash P \text{ then } P \in V(w)
\]
Historical remarks

Handwritten proofs since 1950’s

- ... Sequent calculi [Beth, Gentzen]
- Tableau calculi
  (tableau proof = sequent proof backwards)
- Kripke: explicit accessibility relation
- Smullyan, Fitting: uniform notation
- Single-step tableau [F. Massacci]
  \[ \sigma : \Diamond A \Rightarrow \sigma, n : A \]
- Tableau by graph rewriting [Castilho et al.] ... 

Nowadays mechanized systems

- Fast provers: FaCT [Horrocks], LWB [Heuerding]
  K-SAT [Giunchiglia & Sebastiani]
- Generic provers: TWB [Abate & Goré], LoTREC
Part 2: Practice

3 LoTREC
   - Language
   - Rules
   - Strategies
   - Tableau notation
   - Certifying algorithms

4 Implementation
   - Classical logic
   - Modal logic $K$
   - Multi-modal logic $K_n$
   - $KT \oplus K_n$
   - $S4 \oplus K_n$
   - Suggestions
LoTREC: named after Toulouse-LauTREC
Time-line

- 1999, D. Fauthoux: rewriting kernel + CPL and K
- 2002-2005 M. Sahade: loops in S4, many logics, talking with SAT, completeness and termination general proofs...
- 2006-2009 Me:
  - graph rewriting basis & their theoretical properties,
  - Model checking, LTL, PDL... and other (variants of) logics,
  - One occurrence rule application,
  - Confluence & commutative patterns,
  - Step-by-step run + user interference,
  - GUI, full-web accessibility & performance issues,
  - ...
The black box

Logic Definition

Input Formula
  Partial Premodel

LoTREC

Graphs
  Extensible to models
  Not Extensible to models
# User-defined language

## Atomic propositions
- Any constant symbol = Capital 1st letter words

## Formulas
- defined uniformly as Terms (prefix notation)
- displayed in ≠ form (ex. infix notation)

### Example (definition)

<table>
<thead>
<tr>
<th>name</th>
<th>arity</th>
<th>display</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>1</td>
<td>~ -</td>
</tr>
<tr>
<td>and</td>
<td>2</td>
<td>- &amp; -</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pos</td>
<td>1</td>
<td>&lt;&gt; -</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Example (usage)

- pos P
  - <> P
- and not Q not P
  - ~ Q & ~ P
On paper

Truth conditions

+ Structural constraints

$M, w \models A \land B$ iff

$M, w \not\models A$ and $M, w \not\models B$

as

Graph rewriting rules

\[
\text{A} \& \text{B} \\
\text{A} \\
\text{B}
\]
On paper

Truth conditions

+ Structural constraints

as Graph rewriting rules

\[ M, w \vdash \Diamond A \text{ iff } \exists u \mid wRu \text{ and } M, u \vdash A \]
On paper

Truth conditions + Structural constraints as Graph rewriting rules

Model is reflexive
In LoTREC

Graph rewriting rule as “if Conditions ... then Actions”

Rule And

hasElement node and variable A variable B

add node variable A
add node variable B

End
In LoTREC

Graph rewriting rule as “if Conditions ... then Actions”

Rule  Pos

hasElement  node1 pos  variable  A

createnewnode  node2

link  node1  node2  R

add  node2  variable  A

End
Graph rewriting rule as “if Conditions ... then Actions”

**Rule** ReflexiveArcs

```plaintext
isNewNode node

link node node R
End
```
Informal definition

Apply rule = apply to every formula in every node (unless..)

→ strategies get more declarative
→ proofs get easier

Tableau rules expand directed graphs by

- adding links
- adding nodes
- adding formulas
- duplicating the graph

\[
\text{rule}(G) = \{ G_1, \ldots, G_n \}
\]

\[
\text{rule}(\{ G_1, \ldots, G_n \}) = \text{rule}(G_1) \cup \ldots \cup \text{rule}(G_n)
\]
Non-determinism
Non-determinism
Non-determinism
Non-determinism
Non-determinism

Premodel$_{1}$  Premodel$_{2}$  Premodel$_{n}$
Why a strategy?

- An order of rules application:
  - **Strategy** CPL
    - Stop
    - And
    - Or...

- **Saturation:**
  - **Strategy** K
    - repeat
    - CPL
    - Pos
    - Nec
    - end
Semantics

- **block**: `rule1 ... rulen ... anotherStrategy ...`
- apply all applicable rules in **order** then **stops**

**Example**

**Strategy** CPL

- Stop
- And
- Or
- Not_Not

...
Semantics

- block: `rule1 ... ruleN ... anotherStrategy ...`
  apply all applicable rules in order then **stops**

- **repeat** block **end**
  repeat until no rule applicable (**saturation**)

Example

**Strategy**  K

```
repeat
  CPL
    Pos
    Nec
end
```

In most cases: **repeat** and blocks are sufficient!
Semantics

- **block**: `rule1 ... rule n ... anotherStrategy ...` apply all applicable rules in **order** then **stops**

- **repeat** block **end**
  repeat until no rule applicable (**saturation**)

- **firstRule** block **end**
  apply first applicable rule, then stop (**unfair!**)

Example

```c
repeat
  firstRule
  rule1
  rule2  x
  end
end
```

`rule1` is always applicable
`rule2` is applicable
BUT never applied!
Semantics

- **block**: `rule1 ... ruleren ... anotherStrategy ...`
  apply all applicable rules in **order** then **stops**

- **repeat block end**
  repeat until no rule applicable (**saturation**)

- **firstRule block end**
  apply first applicable rule, then stop (**unfair!**)

- **allRules block end**
  exactly as a “**block**”, but needed inside **firstRule**

**Example**

```
[89x107] firstRule
rule1
[101x80] allRules
[113x67] rule2
[113x53] rule3
end
[89x12] end
```

[33 / 56]
Semantics

- block: `rule1 ... rulesn ... anotherStrategy ...`
  apply all applicable rules in order then stops

- `repeat block end`
  repeat until no rule applicable (saturation)

- `firstRule block end`
  apply first applicable rule, then stop (unfair!)

- `allRules block end`
  exactly as a “block”, but needed inside `firstRule`

- `applyOnce rule`
  apply the rule on only one occurrence
Tableau definition

The set of tableaux for $A$ with strategy $S$ is the set of graphs obtained by applying the strategy $S$ to an initial single-node graph whose root contains only $A$.

- Notation: $S(A)$

Remark

our tableau = “tableau branch” in the literature (sounds odd to call a graph a branch)
Open or Closed?

- A node is closed iff it contains “False” (unless..)
- A tableau is closed iff it has a closed node
- A set of tableaux is closed iff all its elements are closed

An open tableau is a premodel

⇒ build a model
To be proved for each strategy $S$!

- **Termination**
  For every $A$, $S(A)$ terminates.

- **Soundness**
  If $S(A)$ is closed then $A$ is unsatisfiable.

- **Completeness**
  If $S(A)$ is open then $A$ is satisfiable
In general...

- Soundness proofs: easy (we apply truth conditions)
- Termination proofs: not so easy (case-by-case)
- Completeness proofs...
  - ...for fair strategies: standard techniques work “in most cases”
    but fair strategies do not terminate in general
  - ...for terminating strategies: difficult
    rigorous proofs rare even for the basic modal logics!
    reason: strategy = imperative programming
**In general...**

BUT soundness + termination is practically sufficient (e.g. when experimenting with a logic):

- apply strategy to $A$
- take an open tableau and build pointed model $(M, w)$
- check if $M$ in desired class of models
- check if $M, w \models A$
A general termination theorem

[O. Gasquet et al., AIML 2006]

- IF for every rule $\rho$:
  - the RHS of $\rho$ contains **strict** subformulas of its LHS
  - AND
  - some restriction on node creation

- THEN
  - for every formula A:
  - the tableaux construction terminates
Another general termination theorem

[O. Gasquet et al., AIML 2006]

- IF for every rule $\rho$:
  - the RHS of $\rho$ contains subformulas of its LHS
  - AND
  - some restriction on node creation
  - AND
  - some loop testing in the strategy

- THEN
  - for every formula $A$:
  - the tableaux construction terminates
Termination from graph rewriting

Similarly, undecidable. . . BUT:

- **Forward closures** [D. Plump]
  - delete at least one element of a pattern on which a rule was applied ⇒ do not apply a rule twice on the same pattern ⇒ guaranteed by default in LoTREC

- **Layered graph grammars** [G. Taentzer]
  - rules of one strata $i$: once applied, go to next strata $i+1$ & never back again to $i$ . . .

. . .
Part 2: Practice

3 LoTREC
- Language
- Rules
- Strategies
- Tableau notation
- Certifying algorithms

4 Implementation
- Classical logic
- Modal logic $K$
- Multi-modal logic $K_n$
- $KT \oplus K_n$
- $S4 \oplus K_n$
- Suggestions
How to get LoTREC

http://www.irit.fr/Lotrec (Capital “L”)

- Launch

- or, Download → Executable to get LoTREC_2.0.zip, so unzip, then run.bat

(I will check install problems with you during the break)
How to proceed

CPL: Classical Propositional Logic

1. From the task pane, open:
   *Open Predefined logic* → *Others* → *CPL*

2. Run with
   *Build Models*

3. Why these results?
   - Predefined formula
   - Predefined Main strategy

4. Review the logic definition: *Connectors, Rules* . . .

5. Change the formula

6. Re-run . . .
Adding “↔”

What about formulas with “↔” operator?

1. Save as CPL locally as “CPL_complete.xml”
2. Add to Connectors:
   - Name: equiv
   - Arity: 2
   - Display: _<-->_
   - Priority: 0 (or whatever)
3. Add to Rules:
   - Equiv, and NotEquiv
4. Call them in the strategy
5. Try some formulas...
Algorithm

Classical logic +

- for every $\Diamond A$ at every node $w$
  create a successor $u$ and add $A$ to it
- for every $\Box A$ at every node $w$, and for every successor $u$
  add $A$ to $u$
- transform $\neg\Diamond A$ into $\Box\neg A$
- transform $\neg\Box A$ into $\Diamond\neg A$
- Strategy: saturation with all the rules...
Hints

- **Rule** Pos
  
  `hasElement w pos variable a`
  
  `createNewNode u`
  
  `link w u R`
  
  `add u variable a`

- **Rule** Nec
  
  `hasElement w nec variable a`
  
  `isLinked w u R`
  
  `add u variable a`
How to proceed

1. Continue with your “CPL_complete.xml”, or
   *Open Predefined logic → Others → CPL_complete*

2. Add the necessary connectors and rules

3. Call NotNec and NotNec in CPLStrategy

4. Create a new strategy KStrategy which calls repeatedly CPLStrategy, and the rules Pos and Nec

5. Test with
   and nec P and pos Q pos or R not P
   i.e. [] P & <> Q & <> (R v ¬ P)

6. Play with other formulas...
From $K$ To $K_n$

- Replace the connector $\Diamond_-$ by $\langle \_ \rangle_-$
- Change all the predefined formulae 😄
- Change the modal rules: $Pos$, $Nec$, $NotPos$ and $NotNec$

**Rule** Pos\_K

hasElement w pos variable a

createNewNode u
link w u R
add u variable a

**Rule** Pos\_Multi\_K

hasElement w pos variable r variable a

createNewNode u
link w u variable r
add u variable a
How to proceed

1. From the task pane, open: *Open Predefined logic* → *Others* → *Multi K*
2. Play with formulas . . .
From $K_n$ To $KT \oplus K_n$

The idea is to have:

- Axiom T for “$R1$”: $[R1]P \rightarrow P$ as valid

Or:

- Reflexive models on “$R1$”
From $K_n$ To $KT \oplus K_n$

1. Save $Multi\ K$ as $T + Multi\ K$
2. Add the following rule
   
   **Rule** NecT_R1
   
   ```
   hasElement\ node\ nec\ R1\ variable\ a
   add\ node\ variable\ a
   ```

   Or:

   **Rule** Reflexive_arcs_for_R1
   
   ```
   isNewNode\ w
   link\ w\ w\ R1
   ```

3. Call it in the strategy
4. Play …
From $\text{KT} \oplus K_n$ To $\text{S4} \oplus K_n$

Recall: $\text{S4} = \text{KT}4$

The idea is to have:

- Axiom 4 for "$R1$": $[R1]P \rightarrow [R1][R1]P$ as valid
  
  + Axiom T for "$R1$": $[R1]P \rightarrow P$ as valid

Or:

- Transitive models on "$R1$"
  
  + Reflexive models on "$R1$"
From KT ⊕ K_n To S4 ⊕ K_n

1. Save as again...
2. Copy/Paste \( \text{Nec} \), and rename it as \( \text{Nec4}_R1 \)
3. Change it to the following:

**Rule** Nec4_R1

\[
\text{hasElement} \ \text{node} \ \text{nec} \ R1 \ \text{variable} \ \text{a} \\
\text{isLinked} \ \text{node} \ \text{node}' \ R1
\]

\[
\text{add} \ \text{node}' \ \text{nec} \ R1 \ \text{variable} \ \text{a}
\]

Or:

**Rule** Transitive_arcs_for_R1

\[
\text{isLinked} \ \text{w1} \ \text{w2} \ R1 \\
\text{isLinked} \ \text{w2} \ \text{w3} \ R1
\]

\[
\text{link} \ \text{w1} \ \text{w3} \ R1
\]

*But be careful with... \[ R1 \langle R1 \rangle P \]*
Debugging $S4 \oplus K_n$

$[R1] \langle R1 \rangle P$ is looping!

Solution:

1. Add the rule `loopTest` (from S4Optimal)

   **Rule** `loopTest`
   
   `isNewNode node'`
   
   `isAncestor node node'` (or `isLinked` with transitive arcs)
   
   `contains node node'` (or `haveSameFormulaSet` for=)

   `mark node' CONTAINED`
   
   `link node' node Loop` (optional)

2. Change **Pos**: do not create successors for loop nodes!

3. Call it in the strategy

4. Run again...

   **PS:** change the rule `Or` too, if interested in performance...
It is up to you...

- S4 with histories,
- S5, K+Universal operator,
- Confluence,
- Model checking,
- LTL, PDL (need Model checking first),
- Intuitionistic logic,
- ...
Thank you!