

LoTREC: Theory and Practice

B. Saïd

O. Gasquet, F. Schwarzentruher, 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

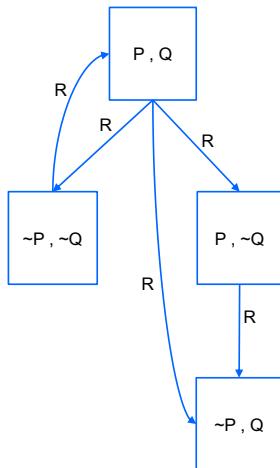
- Possible worlds models
- Constraints on models
- Syntax
- Semantics

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 iff u is possible given the actual world w
- Temporal
 wRu iff u is in the future of w
- Epistemic
 wR_lu iff u is possible for agent l , given the actual world w
- Dynamic
 wR_Pu iff u is a possible result of the execution of the program / action P in w

Readings of $R \Rightarrow$ Properties of R

Readings of “ R ”

- Alethic
 wRu iff u is possible given the actual world w
- Temporal
 wRu iff u is in the future of w
- Epistemic
 wR_lu iff u is possible for agent l , given the actual world w
- Dynamic
 wR_Pu iff u is a possible result of the execution of the program / action P 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 $op(a_1, \dots, a_n)$
 - $Bel_I A$ 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
 - $\square A$ A is **necessary** (true in all possible worlds)
 - $\diamond A$ A is **possible**
- In general:
 - $\diamond A \Leftrightarrow \neg \square \neg A$
 - $\square A \Leftrightarrow \neg \diamond \neg A$

What is a formula?

BNF:

$$A ::= P \mid op(A_1, \dots, A_n)$$

i.e.

$$A ::= P \mid \neg A \mid A \wedge A \mid A \vee A \mid \diamond A \mid \square A \mid \langle A \rangle A \mid [A]A \mid K_A A \dots$$

Example

- $P \wedge \neg Q \wedge \square Q \wedge \diamond (P \wedge \square \neg Q)$
- $Door_Closed \wedge [Open] Door_Open$
- $Card_Is_Red \wedge K_{Ann} Card_Is_Red \wedge K_{Ann} \neg K_{Bob} Card_Is_Red$

Truth conditions

■ Atoms

- $M, w \Vdash P$ iff $P \in V(w)$

■ Classical connectives

- $M, w \Vdash A \wedge B$ iff $M, w \Vdash A$ and $M, w \Vdash B$
- ...

■ Non-classical operators

Via the accessibility relation R

- $M, w \Vdash \Diamond A$ iff **exists** $u: wRu$ and $M, u \Vdash A$
- $M, w \Vdash \Box A$ iff **for all** $u: wRu$ then $M, u \Vdash A$

Truth conditions

■ Multi-modal operators

- $M, w \Vdash \langle I \rangle A$ iff exists $u: wR_I u$ and $M, u \Vdash A$
- ...

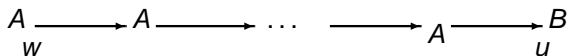
■ Algebraic operators

- $M, w \Vdash \diamond^{-1} A$ iff exists $u: wR^{-1} u$ and $M, u \Vdash A$
- $M, w \Vdash \langle I \cup J \rangle A$ iff exists $u: w(R_I \cup R_J) u$ and $M, u \Vdash A$
- $M, w \Vdash \langle I^* \rangle A$ iff exists $u: wR_I^* u$ and $M, u \Vdash A$

Truth conditions

■ Temporal operators

- $M, w \Vdash \Box A$ iff exists $u: wRu$ and $M, u \Vdash A$
- $M, w \Vdash \Diamond A$ iff exists $n, u: wR^n u$ and $M, u \Vdash A$



- $M, w \Vdash AUB$ iff exists $u: wR^* u$ and $M, u \Vdash B$
and for all $v : (wR^* v$ and $vR^+ u)$, $M, v \Vdash A$
- ...

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 = 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

- $\Box(P \vee \neg P)$
- $\Box(P \wedge Q) \rightarrow \Box P \wedge \Box Q$
- $\Box P \wedge \Box Q \rightarrow \Box(P \wedge Q)$

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

Example

- P
- $P \wedge \neg \Box P$
- $P \wedge \Box \neg P$
- $\Box P \wedge \neg \Box \Box P$

Other Classes

\mathcal{C} = some subset of K

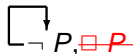
- A is **valid** in \mathcal{C} iff **for all** M in \mathcal{C} and **all** w in M : $M, w \Vdash A$

Example

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

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

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



- A is **satisfiable** in \mathcal{C} iff **for some** M in \mathcal{C} and **some** w in M : $M, w \Vdash A$

Example

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

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

A is valid in \mathcal{C} iff $\neg A$ is unsatisfiable in \mathcal{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 $M, w \Vdash A$?

- Validity

Given A and \mathcal{C} is A valid in \mathcal{C} ?

- Satisfiability

Given A and \mathcal{C} does there exist M in \mathcal{C} and w in M :
 $M, w \Vdash A$?

- Model construction

Given A and \mathcal{C} **compute** M in \mathcal{C} and w in M :
 $M, w \Vdash A$

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)

- $M, w \Vdash A \wedge B \Rightarrow$ add $M, w \Vdash A$ and add $M, w \Vdash B$
- $M, w \Vdash A \vee B \Rightarrow$ add either $M, w \Vdash A$ or add $M, w \Vdash B$
(non det.)
- $M, w \Vdash \neg A \Rightarrow$ don't add $M, w \Vdash A$!!
 - $M, w \Vdash \neg\neg A \Rightarrow$ add $M, w \Vdash A$
 - $M, w \Vdash \neg(A \vee B) \Rightarrow$ add $M, w \Vdash \neg A$ and add $M, w \Vdash \neg B$
 - $M, w \Vdash \neg(A \wedge B) \Rightarrow$ 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)

- $M, w \Vdash A \wedge B \Rightarrow$ add $M, w \Vdash A$ and add $M, w \Vdash B$
- $M, w \Vdash A \vee B \Rightarrow$ add either $M, w \Vdash A$ or add $M, w \Vdash B$
(non det.)
- $M, w \Vdash \neg A \Rightarrow$ don't add $M, w \Vdash A$!!
 - $M, w \Vdash \neg\neg A \Rightarrow$ add $M, w \Vdash A$
 - $M, w \Vdash \neg(A \vee B) \Rightarrow$ add $M, w \Vdash \neg A$ and add $M, w \Vdash \neg B$
 - $M, w \Vdash \neg(A \wedge B) \Rightarrow$ 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)
 $W = \{w\}, R = \emptyset, V(w) = \{P : M, w \Vdash P\}$

Modal logic

Basic cases

- $M, w \Vdash \Diamond A$
 - ➔ add some new node u , add wRu , add $M, u \Vdash A$
- $M, w \Vdash \Box A$
 - ➔ for all node u s.t. wRu , add $M, u \Vdash A$

Apply truth conditions = Build a labeled graph

- creates nodes
- add links
- add formulas to nodes

Example

a **node** with the **input formula**

$$\Box P \ \& \ \langle \rangle Q \ \& \ \langle \rangle (R \vee \sim P)$$

Example

$M, w \Vdash A \wedge B$ iff $M, w \Vdash A$ and $M, w \Vdash B$

A is $\Box P$

B is $\Diamond Q \wedge \Diamond(R \vee \neg P)$

$\Box P \ \& \ \Leftrightarrow \ \Diamond Q \ \& \ \Leftrightarrow \ (R \vee \sim P)$

Example

$M, w \Vdash A \wedge B$ iff $M, w \Vdash A$ and $M, w \Vdash B$

A is $\Box P$

B is $\Diamond Q \wedge \Diamond(R \vee \neg P)$

$\Box P \ \& \ \Leftrightarrow \ \Diamond Q \ \& \ \Leftrightarrow \ (R \vee \sim P)$

$\Box P$

$\Leftrightarrow \ \Diamond Q \ \& \ \Leftrightarrow \ (R \vee \sim P)$

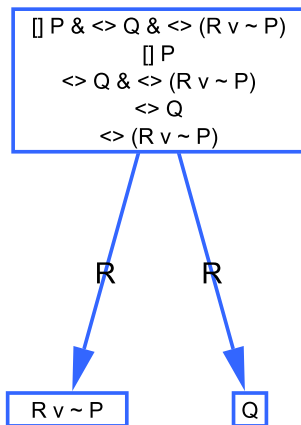
Example

$M, w \Vdash A \wedge B$ iff $M, w \Vdash A$ and $M, w \Vdash B$

$\begin{aligned} & \Box P \ \& \ \langle \rangle Q \ \& \ \langle \rangle (R \vee \sim P) \\ & \quad \Box P \\ & \quad \langle \rangle Q \ \& \ \langle \rangle (R \vee \sim P) \\ & \quad \quad \langle \rangle Q \\ & \quad \quad \langle \rangle (R \vee \sim P) \end{aligned}$
--

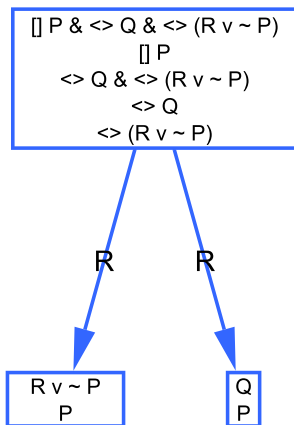
Example

$M, w \Vdash \diamond A$ iff $\exists u \mid wRu$ and $M, u \Vdash A$



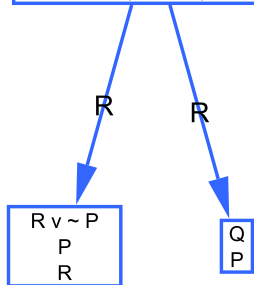
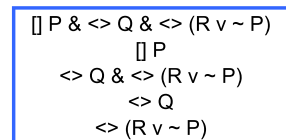
Example

$M, w \Vdash \Box A$ iff $\forall u : wRu$ then $M, u \Vdash A$

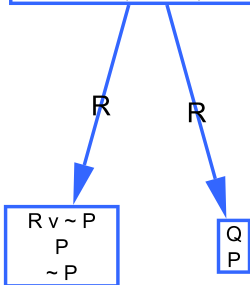
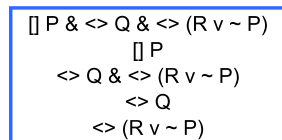


Example

$M, w \Vdash A \vee B$ iff $M, w \Vdash A$ or $M, w \Vdash B$

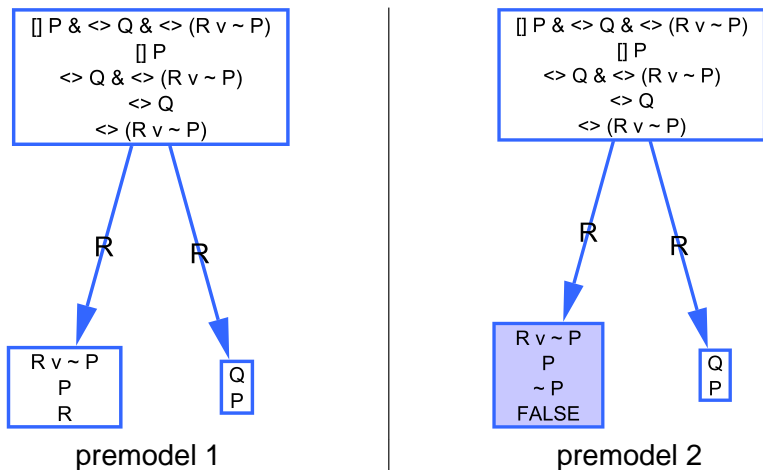


premodel 1

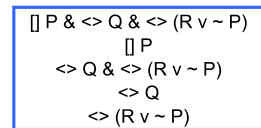


premodel 2

Example

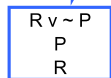


Example



R

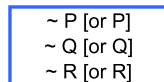
R



premodel 1

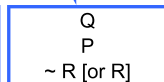
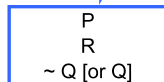


Model
 \implies
extraction



R

R


 $M, w \Vdash P \text{ then } P \in V(w)$

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

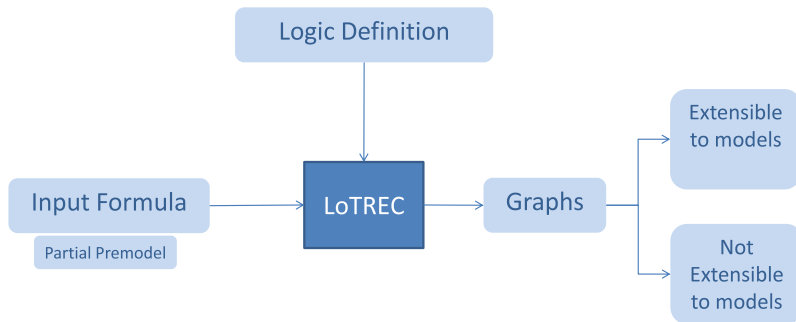
LoTREC: named after Toulouse-LautREC



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



User-defined language

Atomic propositions

- Any constant symbol = Capital_1st_letter_words

Formulas

- defined uniformly as Terms (prefix notation)
- displayed in \neq form (ex. infix notation)

Example (definition)

name	arity	display
not	1	$\sim -$
and	2	$- \& -$
...		
pos	1	$\langle \rangle -$
...		

Example (usage)

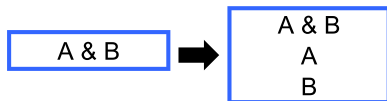
- pos P
 $\langle \rangle P$
- and not Q not P
 $\sim Q \& \sim P$

On paper

Truth conditions
+
Structural constraints

as Graph rewriting rules

$M, w \Vdash A \wedge B$ iff
 $M, w \Vdash A$ and $M, w \Vdash B$

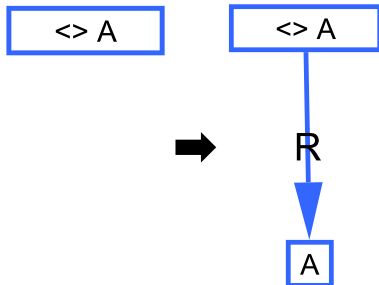


On paper

Truth conditions
+
Structural constraints

as Graph rewriting rules

$M, w \Vdash \diamond A$ iff
 $\exists u \mid wRu \text{ and } M, u \Vdash A$

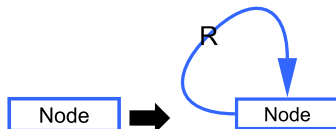


On paper

Truth conditions
+
Structural constraints

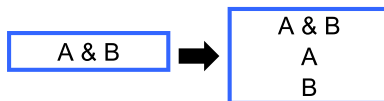
as Graph rewriting rules

Model is reflexive



In LoTREC

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



Rule And

hasElement node and variable A variable B

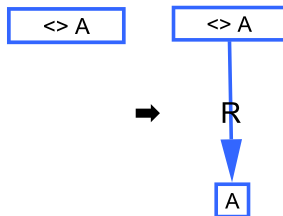
add node variable A

add node variable B

End

In LoTREC

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



Rule Pos

hasElement node1 pos variable A

createNewNode node2

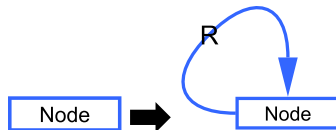
link node1 node2 R

add node2 variable A

End

In LoTREC

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



Rule ReflexiveArcs

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

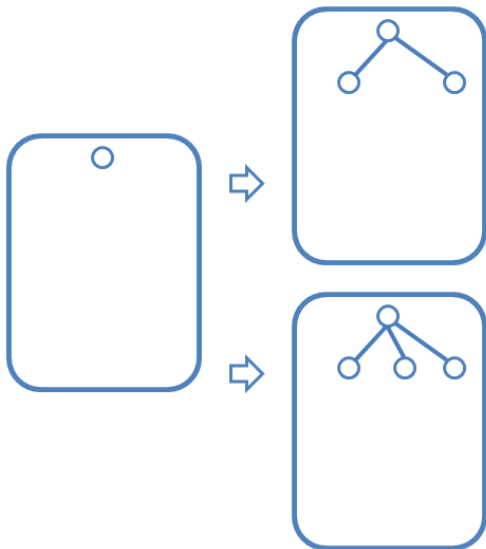
$$rule(G) = \{G_1, \dots, G_n\}$$

$$rule(\{G_1, \dots, G_n\}) = rule(G_1) \cup \dots \cup rule(G_n)$$

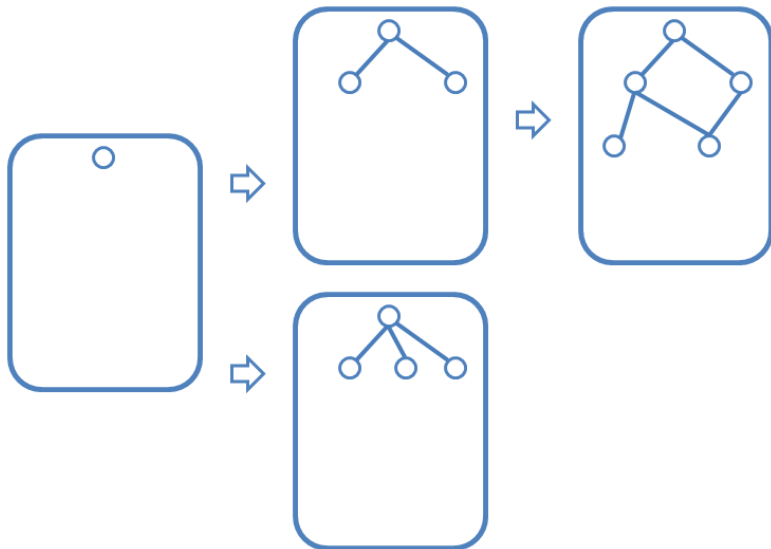
Non-determinism



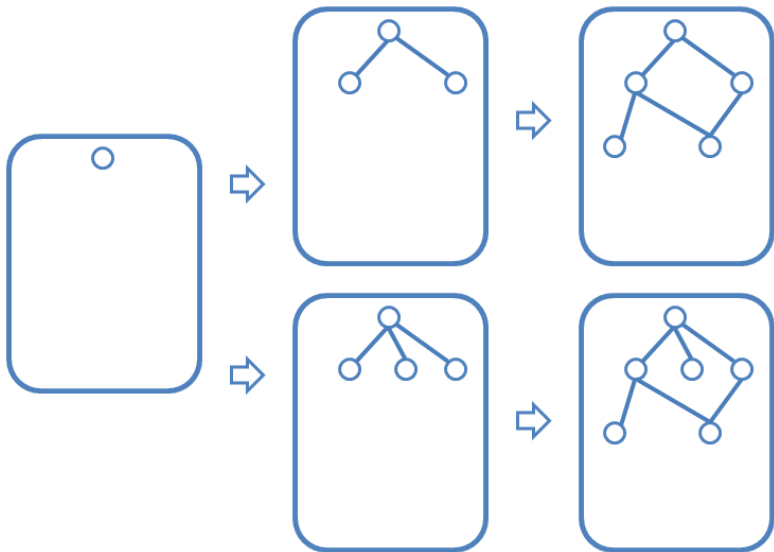
Non-determinism



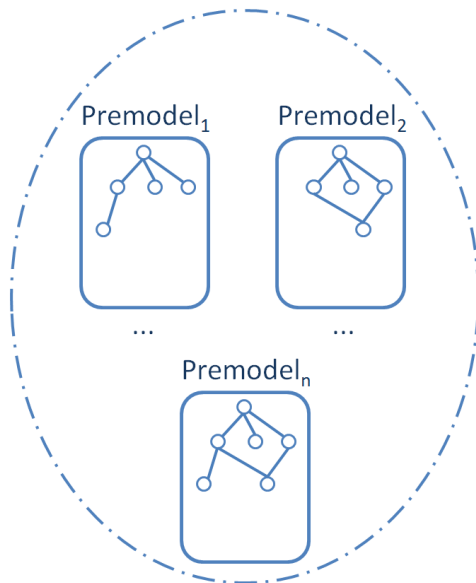
Non-determinism



Non-determinism



Non-determinism



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 \mathcal{K}

repeat

CPL

Pos

Nec

end

In most cases: repeat and blocks are sufficient!

Semantics

- **block**: rule1 ... rulen ... 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

```
repeat  
  firstRule  
    rule1  
    rule2  X  
  end  
end
```

rule1 is always applicable
rule2 is applicable
BUT never applied!

Semantics

- **block**: rule1 ... rulen ... 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

```
firstRule
  rule1
  allRules
    rule2
    rule3
  end
  rule4
end
```

Semantics

- **block**: rule1 ... rulen ... 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
 \implies build a model

Formal properties

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 $M, w \Vdash A$

A general termination theorem

[O. Gasquet et al., AIML 2006]

- IF for every rule ρ :
 - the RHS of ρ 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 ρ :
 - the RHS of ρ 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 \Rightarrow do not apply a rule twice on the same pattern
 \Rightarrow 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 \rightarrow *Others* \rightarrow *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 “ \leftrightarrow ”

What about formulas with “ \leftrightarrow ” operator?

1 Save as *CPL locally* as “*CPL_complete.xml*”

2 Add to *Connectors*:

name	arity	display	priority
equiv	2	$_ \leftrightarrow _$	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 \rightarrow *Others* \rightarrow *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. $\Box P \ \& \ \langle \rangle Q \ \& \ \langle \rangle (R \vee \sim P)$
- 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 \rightarrow *Others* \rightarrow *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 $T + \text{Multi K}$

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 $KT \oplus K_n$ To $S4 \oplus K_n$

Recall: $S4 = KT4$

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 $KT \oplus K_n$ To $S4 \oplus K_n$

- 1 Save as again...
- 2 Copy/Paste Nec , and rename it as $Nec4_R1$
- 3 Change it to the following:

Rule $Nec4_R1$

hasElement node nec R1 variable a
isLinked node node' R1

add node' nec R1 variable a

Or:

Rule $Transitive_arcs_for_R1$

isLinked w1 w2 R1

isLinked w2 w3 R1

link w1 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!