Graph Rewriting for Model Construction in Modal Logic

Bilal SAID

Institut de Recherche en Informatique de Toulouse - IRIT
Université de Toulouse

29 January 2010
## Locating my work

<table>
<thead>
<tr>
<th>Automated reasoning</th>
<th>Theory</th>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Graph rewriting</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Panorama

1 Modal logics

2 Model Construction

3 Event-driven pattern matching
Outline

1 Modal logics
   - Modelling with graphs
   - Talking about graphs
   - Reasoning about graphs

2 Model Construction

3 Event-driven pattern matching
Example: Switch-Bulb system
Example: Switch-Bulb system
Various systems, situations...

Guess-the-card game

Traffic-light

...
Various systems, situations...
Switch-Bulb system

at the state "Light_Off", after "Toggle" necessarily "Light_On"
Switch-Bulb-Mouse
Switch-Bulb-Mouse
at state “Light_Off”, after “Toggle” not necessarily “Light_On”
at state “Light_Off”, after “Toggle” not necessarily “Light_On”
i.e. possibly not “Light_On”
at state “Light_Off”, after “Toggle” not necessarily “Light_On”
i.e. possibly not “Light_On”
i.e. possibly “Light_Off”
Formal language: Formulas

- “Light_Off” and after “Toggle” necessarily “Light_On”
  \[ \text{Light\_Off} \land [\text{Toggle}] \text{Light\_On} \]

- “Light_Off” and after “Toggle” possibly “Light_Off”
  \[ \text{Light\_Off} \land \langle \text{Toggle} \rangle \text{Light\_Off} \]

Generic sentences:

- Necessarily \( A \) \( \square A \)
- Possibly \( A \) \( \diamond A \)

How to evaluate these formulas?
Formal language: Formulas

- "Light_Off" and after "Toggle" necessarily "Light_On"
  \[
  \text{Light}_{\text{Off}} \land [\text{Toggle}] \text{Light}_{\text{On}}
  \]

- "Light_Off" and after "Toggle" possibly "Light_Off"
  \[
  \text{Light}_{\text{Off}} \land \langle \text{Toggle} \rangle \text{Light}_{\text{Off}}
  \]

Generic sentences:

Necessarily $A$ $\Box A$
Possibly $A$ $\Diamond A$

How to evaluate these formulas?
Formal language: Formulas

- “Light_Off” and after “Toggle” necessarily “Light_On”
  \[ \text{Light\_Off} \land [\text{Toggle}] \text{Light\_On} \]

- “Light_Off” and after “Toggle” possibly “Light_Off”
  \[ \text{Light\_Off} \land \langle \text{Toggle} \rangle \text{Light\_Off} \]

Generic sentences:
- Necessarily \( A \) \( \square A \)
- Possibly \( A \) \( \Diamond A \)

How to evaluate these formulas?
Kripke models

Model
= transition system

- possible Worlds
  = states

- accessibility Relation
  = transitions

- Valuation
  = labeling function

\[ M = (W, R, V) \]
Semantics: Truth conditions

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

- **Classical operators**
  - $M, w \models A \land B$ iff $M, w \models A$ and $M, w \models B$
  - $M, w \models A \lor B$ iff $M, w \models A$ or $M, w \models B$
  - ...  

- **Modal operators**
  - $M, w \models \Box A$ iff for all $u$, if $wRu$ then $M, w \models A$
  - $M, w \models \Diamond A$ iff exists $u$ s.t. $wRu$ and $M, u \models A$
Example

- $M, u \models \text{Light}_\text{On}$
- $M, w \models \text{Light}_\text{Off} \land [\text{Toggle}]\text{Light}_\text{On}$
- $\ldots$
Various systems, situations...

Examples

- The card is red, Alice knows it and knows that Bob does not:
  \[ \text{Card}_\text{Red} \land K_{\text{Alice}} \text{Card}_\text{Red} \land K_{\text{Alice}} \neg K_{\text{Bob}} \text{Card}_\text{Red} \]

- Always, if it is Red then next it turns out Green:
  \[ G (\text{Red} \rightarrow X \text{Green}) \]

- ...

How to take these into account?

- Change truth conditions
- Constraints on \( R \)
- Constraints on \( V \)
Various systems, situations... 

Examples

- The card is red, Alice knows it and knows that Bob does not know it:
  \[ \text{Card}_\text{Red} \land K_{\text{Alice}}\text{Card}_\text{Red} \land K_{\text{Alice}}\neg K_{\text{Bob}}\text{Card}_\text{Red} \]
- Always, if it is Red then next it turns out Green:
  \[ G(\text{Red} \rightarrow X\text{Green}) \]
- ...

How to take these into account?

- Change truth conditions
- Constraints on \( R \)
- Constraints on \( V \)
Constraints on $R$

One relation:

- Transitive
  
  future of future is future

- Reflexive
  
  I know s.th. hence it is true

- Serial
  
  there is always a future

- 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

- ...

Confluent (Church-Rosser)
Modelling with graphs

Talking about graphs

Reasoning about graphs

Constraints on $R$

One relation:
- Transitive
  - future of future is future
- Reflexive
  - I know s.th. hence it is true
- Serial
  - there is always a future
- 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 $R$

One relation:
- Transitive
  - \textit{future of future is future}
- Reflexive
  - \textit{I know s.th. hence it is true}
- Serial
  - \textit{there is always a future}
- Symmetric
- Equivalence (\textit{universal})
- Confluent (\textit{Church-Rosser})
- \ldots

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
- \ldots
One relation:
- Transitive
  - future of future is future
- Reflexive
  - I know s.th. hence it is true
- Serial
  - there is always a future
- 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 $V$

- HL(∀): a nominal is true at a unique world
- Intuitionistic: atoms persist along paths
- PAL: literals persist throughout (updated) models
- ...
## Interests

<table>
<thead>
<tr>
<th>Given</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property P</td>
<td>does S have P?</td>
</tr>
<tr>
<td>System S</td>
<td>is there a system S having P?</td>
</tr>
<tr>
<td>Property P</td>
<td>which system S may have P?</td>
</tr>
</tbody>
</table>
# Problems

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model Checking</strong></td>
<td>Yes/No</td>
</tr>
<tr>
<td>Model Checking</td>
<td>×</td>
</tr>
<tr>
<td><strong>Satisfiability / Validity</strong></td>
<td>Yes/No</td>
</tr>
<tr>
<td>Satisfiability / Validity</td>
<td>×</td>
</tr>
<tr>
<td><strong>Model Construction</strong></td>
<td>Model/Counter-Model</td>
</tr>
<tr>
<td>Model Construction</td>
<td>×</td>
</tr>
</tbody>
</table>
Methods

Since 1950’s...

- Sequent calculi [Beth, Gentzen, ...]
  - Proof search

- Tableau calculi [Smullyan, Fitting, Massacci, ...]
  - Model existence check

- Tableau by graph rewriting [à la toulousaine]
  - Model construction
**Example**

A node with the input formula

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

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

A

B

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

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

\[
\begin{align*}
\Box P & \land \Diamond Q \land \Diamond (R \lor \neg P) \\
\Box P \\
\Diamond Q & \land \Diamond (R \lor \neg P)
\end{align*}
\]
Modelling with graphs

Talking about graphs

Example

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

\[
\begin{array}{c}
\langle\langle P & \& \langle\langle Q & \& \langle\langle (R \lor \sim P) \\
\langle\langle P \\
\langle\langle Q & \& \langle\langle (R \lor \sim P) \\
\langle\langle Q \\
\langle\langle (R \lor \sim P)
\end{array}
\]
Example

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

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

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

Premodel 1

Premodel 2
Example

premodel 1

\[ \Box P \land \langle \Box Q \land \langle R \lor \sim P \rangle \rangle \]

\[ \Box P \]

\[ \langle \Box Q \land \langle R \lor \sim P \rangle \rangle \]

\[ \langle R \lor \sim P \rangle \]

\[ R \]

\[ R \]

\[ R \]

\[ R \]

\[ R \]

\[ Q \]

\[ Q \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]

\[ P \]
Reasoning Tools

- Fast (but: geek!, not generic)
  - FaCT [Horrocks]
  - LWB [Heuerding]
  - K-SAT [Giunchiglia & Sebastiani]

- Generic (but: limited, requires coding in OCaml)
  - TWB [Abate & Goré]

- Educative (but: not generic!)
  - OOPS [Valkenhoef, Vaart & Verbrugge]
  - Molle [Mazzucchi & Mocci]
Reasoning Tools

Molle

Reflexivity only!
Reasoning Tools

OOPS

- Model update
- High-level language
- Graph viewer

- $S5_n$ only!

Update

$\Rightarrow$
Desiderata

- Generic: users’ own new methods
- Educative: user-friendly
- Performance: reasonable time

Target Users

- Researchers
- Logic
- Computer Science
- Students
- Philosophy
Outline

1. Modal logics

2. Model Construction
   - Graph rewriting rules
   - On paper
   - In LoTREC
   - In Demo

3. Event-driven pattern matching
Uniform methodology

Semantics

via

Graph rewriting rules

How?

What are graph rewriting rules?
Uniform methodology

Semantics

via

Graph rewriting rules

How?

What are graph rewriting rules?
Definition

Graph rewriting rules

On paper

In LoTREC

In Demo

Current_State

Is

Then

S1

S2

next

L

Current_State

Is

Then

S1

S2

R
Graph rewriting rules

Matching

\[ G \]

Current State

\[ \text{Is} \quad \text{Then} \quad \text{next} \quad \text{Is} \]

\[ S_1 \quad S_2 \]

\[ m \]

Current State

\[ \text{Is} \]

\[ \text{Red} \quad \text{Then} \quad \text{Green} \]

\[ \text{Then} \quad \text{Then} \quad \text{Then} \]

\[ \text{Yellow} \]
Application

Graph rewriting rules

On paper
In LoTREC
In Demo

Current_State

Is

next

S1

Then

S2

m

Current_State

Is

m'

S1

Then

S2

next'

Red

Then

Green

Then

Red

Then

Green

Then

Yellow

G

H
Uniform methodology

Semantics

via

Graph rewriting rules

How?

What are graph rewriting rules?
Truth conditions

\[ M, w \Vdash \Diamond A \text{ iff } \exists u \mid wRu \text{ and } M, u \Vdash A \]
**R-Constraints**

Transitivity

Graph rewriting rules

On paper

In LoTREC

In Demo
V-Constraints

Persistence

\[ \begin{array}{c}
\text{P} \\
\text{R} \\
\text{P}
\end{array} \quad \text{\rightarrow} \quad \begin{array}{c}
\text{P} \\
\text{R} \\
\text{P}
\end{array} \]
Certifying the method

- Termination: does it halt?
- Soundness: does it consider ALL the semantics?
- Completeness: does it consider s.th. ELSE?
- Complexity: what is its time / space costs?
- ...

How to experiment with it?
User-defined language

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

Graph rewriting rules

On paper

In LoTREC

In Demo
User-defined rules

Rule Pos

hasElement w pos variable A

createNewNode u

link w u R

add u variable A

End
User-defined rules

Rule Pos

isLinked w u R

isLinked u v R

link w u R

End
User-defined strategies

- Ordering on the rules
- Saturation (*repeat...end*)
- Priority (*firstRule...end*)
The black box

Input Formula → LoTREC → Graphs

Logic Definition

Extensible to models
Not Extensible to models
What was done?
What was done?

The Formula Must Begin by the Name of the Strategy!!!

Formula: $K\text{strategy and and (not and A not B) nec A not nec B}$

Run

Graphic Mode Text Mode Graphic and Text Mode

OutPut
Some of what I did
Some of what I did

Added techniques:
- One occurrence rules application (K.alt, LTL,...)
- Defining non-injective patterns (Confluence)
- Extend the language (Model Checking,...)
- Model checking (LTL, PDL,...)
- Labeled formulas (PAL)
- Nodes as memory cells (S4 + histories)
- Run in step-by-step mode (debugging)
- Code new extensions
- ...
Some of what I did
Running live

http://www.irit.fr/Lotrec
Outline

1. Modal logics

2. Model Construction

3. Event-driven pattern matching
   - Mechanism
   - Semantics
   - Evaluation
Naive pattern matching

C1

- hasElement  n  ♠ A
Mechanism Semantics Evaluation

Naive pattern matching

C1
• hasElement  n  A

...
Naive pattern matching

C1

- hasElement n ♦ A
Naive pattern matching

C1

- hasElement n ⚫ A

...
Naive pattern matching

C1
- hasElement n △ A

...
Naive pattern matching

C1

\[
\text{hasElement} \ n \ \Diamond \ A
\]
Naive pattern matching

C1

- hasElement n  ♦ A
Naive pattern matching

C1

• hasElement n 🟦 A
Naive pattern matching

C1
• hasElement n \(\Diamond A\)

\(O(|G|^{|L|})\)
Naive pattern matching

C1

- hasElement n \(\diamond A\)

\[ O(|G|^{|L|}) \]
Naive pattern matching

C1
• hasElement n

\( O(|G|^{|L|}) \)
**Naive pattern matching**

C1

- hasElement n ♠ A

\[ O(|G|^{|L|}) \]
Event-based matching in LoTREC

C1
- hasElement n ◇ A

◇ (p ∧ q)
Event-based matching in LoTREC

C1

• hasElement n ◇ A

◇ (p ∧ q)

Mechanism
Semantics
Evaluation
Event-based matching in LoTREC

C1

- hasElement n ◊ A

◊ (p∧q)

Event

◊ (p∧q)

N
Event-based matching in LoTREC

C1

- hasElement n A

Event

(p\land q) N

(p\land q) N

Mechanism
Semantics
Evaluation
Event-based matching in LoTREC

C1

- hasElement n ◇ A

N

◇ (p∧q)

N

◇ (p∧q)
Event-based matching in LoTREC

$O(k|L|)$ where $k \ll |G|$
Equivalence to usual semantics

[Gasquet, Said & Schwarzentruber 09]

Rewriting with event-based matching

= 

Rewriting without event-based matching

Reasons:

- Every successful pattern is considered
- Only unfruitful events are deleted
Related works

- **PROGRES [Zündorf 99]**
  - chooses an optimal plan over $|L|!$ local search plans
  - tracks invalid patterns

- **Incremental Update [Varró & Varró 04]**
  - tracks successful patterns in DB
  - stores & updates are space & time consuming
VS. other rewriting tools

- Comparison is not fair: formula matching
- Benchmark is hard to setup
- General purpose tools are not competent:
  3 levels confluent graph takes:
  - $\approx 6$ sec in AGG
  - $< 0.5$ sec in LoTREC
VS. naive pattern matching

Nb tentatives of pattern matching
Hardest S4 formulas in LWB benchmark
Conclusion
Research

Implementation of

- Time Sub-Intervals Logic  
  [Goranko et al. 08]

- Public Announcement Logic (PAL)  
  [De Lima et al. 09]

Not possible using another platform!
Academic

Accessed through logic courses:

- **Automated Reasoning**
  Prof. C. Pêcheur
  Université Catholique de Louvain, Belgium

- **FGI 3 - Logik**
  Dr. C. Eschenbach
  University of Hamburg, Germany

- **Logique, informatique et sciences cognitives**
  Prof. R. Villemaire
  University of Quebec at Montreal, Canada
Recap on the contributions of my thesis:

- Develop & maintain the LoTREC platform
- Study & implement new logics
- Promote the software in research & academic fields
- Establish the links with graph rewriting theory
- Study the event-based pattern matching
- Clarify the semantics of our rewriting system
2010: The Odyssey continues...

Currently:

- Book: “Kripke’s World”
  Authors: [Gasquet, Herzig, Said, Schwarzentruber]

Next events:

- Universal Logic 2010, April - Lisbon (tutorial)
- ESSLLI 2010, August - Copenhagen, (1 week course)
What about?

- Language extension: SQL-queries?
- Performance: backtracking?
- Generic interface with other tools: will be user-friendly?
- Converse: what about CPDL?
- New methods for new logics...
Merci!